organized_rnn

Hyeonho Lee

2018.12.22

Contents

RNN이란 2
- 왜 RNNs인가? . . . 2
- Hidden Markov models(HMMs) . . . 2

표기법 2

RNN특징 2

Vanishing or exploding gradients 2

RNN이란

- 순차적으로 들어오는 정보를 처리하는 Neural Network
- 문장에서 다음에 나올 단어를 추측하고 싶은 경우 이전에 나온 단어의 정보가 필요
- 동일한 task를 입력값 시퀀스(x_t)의 모든 요소마다 적용하고, 출력결과 ($\hat{y_t}$)에서는 이전 시간의 계산결과(h_{t-1})에 영향을 받는다.

왜 RNNs인가?

- Non-linear Auto-Regressive with eXogeneous inputs (NARX) model :

$$\hat{y}_{t+1} = F(y_t, t_{t-1}, ..., y_{t-d_y}, x_t, x_{t-1}, ..., x_{t-d_x})$$

  여기서 F는 비선형 함수이다.
- 시계열 자료의 예측에 주로 사용되는 모형으로 Delay(d_y, d_x)을 통해 현재 시점에서 얼마나 먼 과거까지 입력값으로 넣을지 결정
- 고정된 크기의 Delay을 고려하기 때문에 만약 Delay 이전 시점의 정보가 필요한 경우 좋지 않은 결과를 나타냄
- 일반적으로 AR기반 모형은 dependency가 지수적으로 감소하여 Long-range dependencies의 자료에서 사용할 수 없다.

Hidden Markov models(HMMs)

- 은닉변수 x_t의 조건부 확률 분포는 오직 은닉 변수 x_{t-1}에 의존

$$P(X_t|x_0, ..., X_{t-1}) = P(X_t|X_{t-1})$$

- N : 가능한 y의 상태 수
- M : 가능한 x의 상태 수
- N, M을 가정하면 평균에 영향을 주는 M개의 모수와 공분한 행렬에 영향을 주는 $\frac{M(M+1)}{2}$ 개의 모수가 존재하므로 전체 $O(NM^2)$개의 출력모수가 존재한다.
- Long-range dependencies의 자료에서는 M의 값이 너무 커지게 되므로 계산량의 측면에서 비실용적이다.

- 이러한 문제들을 RNN의 경우 HMMs의 계산문제를 피하고 long-range dependecies을 포착하는 모형을 만들 수 있다.

표기법

- x_t : 시간 스텝 t에서의 입력값
- h_t : 시간 스텝 t에서의 중간층(=Hidden state), 네트워크의 메모리 부분으로 이전 시간의 은닉층 값과 현재 시간의 입력값에 의해 계산
- $h_t = f(Ux_t + Wh_{t-1} + b_n)$, 여기서 비선형 함수 f는 보통 tanh or ReLU가 사용되고, $h_0 = 0$으로 초기화시킨다.
- $\hat{y_t}$: 시간 스텝 t에서의 출력값. 보통 $\hat{y_t} = softmax(Vh_t + b_y)$을 사용한다.

RNN특징

- 각 층마다 모수의 값들이 일반적 Deep Neural Network와는 달리 모든 시간 스텝에 대해 모수를 전부 공유하고 있다. 이와 같은 구조를 사용함으로써 학습해야 하는 모수의 수를 줄여준다.
- 해당 시점 이전의 정보를 모두 다 저장하는 중간층 h_t은 네트워크 메모리라고 생각할 수 있다. 하지만 실제 구현에서는 너무 먼 과거에 일어난 일들을 잘 기억하지 못한다. 이를 vanishing gradients problem이라고 한다.
- RNN을 training시키기 위해 시간에 따라 펼쳐진 구조에서 역전파 방법을 이용하여 훈련시킨다. 이와 같은 방법을 backpropagation through time(BPTT)라고 한다.

Vanishing or exploding gradients

- Simple RNN을 수식으로 나타내면 다음과 같다.

$$h_t = f(Ux_t + Wh_{t-1} + b_n)$$

$$\hat{y}_t = softmax(Vh_t + b_y), \quad t = 1, ..., T$$

- 이 때 L을 손실함수라 하면, 모수 W에 대한 gradient는 다음과 같이 구할 수 있다.
- 미분하는과정~
- 앞의 식에서 β을 행렬 norm의 상한(upper bound)이라고 가정하면, ~ 임을 유도할 수 있다.
- 따라서 만약 $|\beta_w \beta_n| < 1$이고, $t - k$가 크면 클수록 미분~~(기울기)가 0이 되고, 이는 해당시점 t에서 먼 과거 k시점의 gradient가 0이 되어 영향을 미치지 않게 되는 것을 의미한다.
- 역으로 exploding gradient인 경우에는 $|\beta_w \beta_h| > 1$인 경우이다.
- Exploding gradient의 경우, gradient 값들이 NaN(not a number)가 되기 때문에 프로그램을 실행시킬 때 쉽게 알아차릴 수 있지만 gradient 값이 0이 되는 경우에는 프로그램 계산상에는 문제가 없기 때문에 알아차리기 힘들다.
- Vanishing gradient 문제 해결 방법
    - W행렬의 초기값 설정 또는 regularization
    - 활성함수 ReLU 사용
    - LSTM or GRU