2014147513 조문기

CRNN은 무엇인가

CRNN의 정의와 그 장점

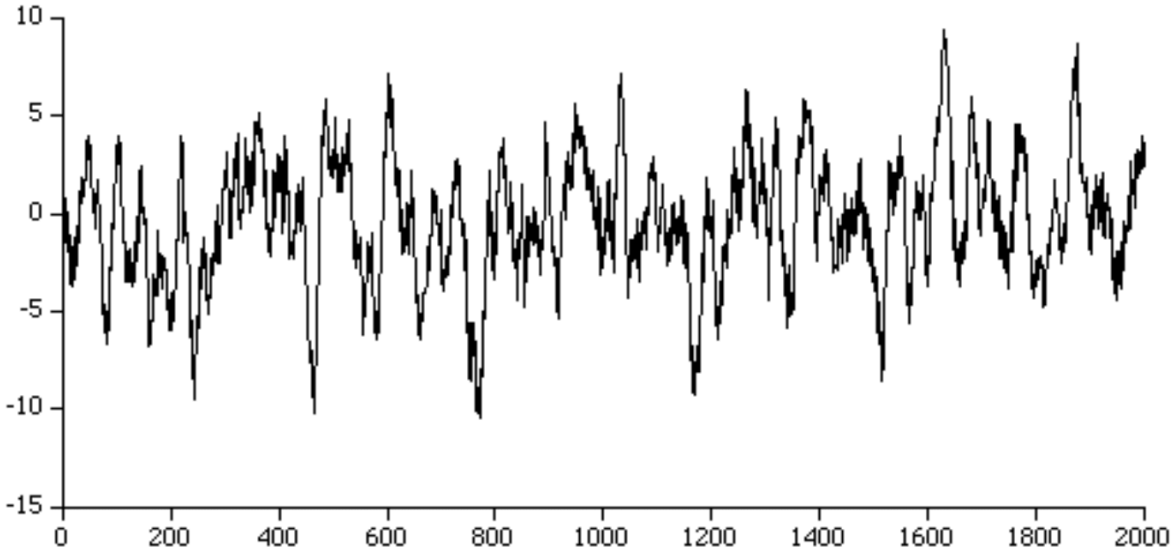
- Sequence Image Processing을 위한 CNN과 RNN의 결합을 CRNN이라고 한다.
- Feature 추출과 시계열 model을 통합시키고 하나의 통일된 framework로 합치는 Neural Network구조를 이야기한다.

1. 기존의 것들이 부분적으로 학습하고 일부를 튜닝하는 것에 비해서 CRNN은 전체를 모델로 한번에 학습이 가능하다. end to end computing
2. 임의의 길이인 시계열 데이터를 다룰 수 있다.
   (No data segmentation || Horizontal scale normalization 등을 포함)
3. 어떤 미리 정해진 어휘에 제한되지 않는다.
4. 사전 free, based분야에서 모두 놀랄만한 성능을 보여준다.
5. 현실에서보다 더 타당한 매우 작은 모델을 효율적으로 만들어낸다.
6. + 이미지 기반 음악 악보 인식에서도 좋은 성능을 보인다. 일반성을 나름 증명하고 있다.

시계열 모델(Time Series Analysis)

- 회귀분석(Linear Regression)분석을 시계열(Time Series)에 적용하는 것을 의미한다.
- 회귀분석은 시점을 고려하지 않으나, 시계열 분석은 시간을 고려한다는 것이 가장 큰 특징이다. 단일 데이터를 가지고 있는 상황에서 이 데이터를 얻은 시간을 알게 된다면, 사실은 1종 데이터가 아니라 2종 데이터가 되는 것이다.
- 많은 학자들은 이 시계열 정보들을 전통적으로 크게 2가지로 나눠 규칙성을 가지는 패턴과 불규칙성을 가지는 패턴으로 나누어 왔다. 그래서 시계열 모형은 전통적으로 이 두가지를 나누어서 개발되어 왔는데, 규칙성을 만드는 패턴을 또한 두가지로 나눠서 이전의 결과와 이후의 결과 사이에서 발생하는 자기상관성(Autocorrelativeness)과 이전에 생긴 불규칙한 사건이 이후의 결과에 편향성을 초래하는 이동 평균(Moving Average)현상으로 구분하고 있다. 많은 시계열 모형들이 불규칙한 패턴을 White Noise라고 칭하고 평균이 0이며 일정한 분산을 지닌 정규분포에서 추출된 임의의 수치라고 규정하고 있다.
- 대표적인 시계열 모델은 AR / MA / ARMA / ARIMA모형이 존재한다.

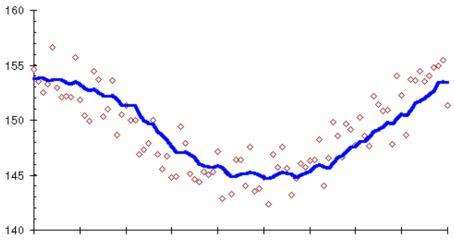
1. AR (AutoRegressive) model
   자기상관이란 어떠한 Random Variable에 대해서 이전의 값이 이후의 값에 대해서 영향을 미치는 것을 의미한다. 이러한 자기상관은 바로 이전의 결과의 영향을 받을 수도 있지만, 드물



게는 Delay가 발생하기도 한다. 나중에 영향을 끼칠 수 있다는 것이다.
대표적인 예로는 스프링이 존재하는데, 이처럼 많은 금융정보나 시계열 정보들이 자기 상관의 특징을 잘 가지고 있다. 이 데이터는 평균이 0 이라는 점 이외에도 이전에 양수가 나오면 그 후는 음수가 나올 것이라는 일정한 패턴도 예측이 가능하다. 이러한 자기상관성을 시계열 모형으로 구성한 것을 AR 모형이라고 부르는데, 가장 간단한 형태가 직전에 이야기한 데이터가 다음 데이터에 영향을 준다고 가정한 AR(1) 모형이다.
X(t) = a * X(t - 1) + c + u * e(t)
시점 t에서 얻게 될 X(t)의 평균값은 시점 t - 1에서 얻었던 X(t - 1)의 값에 a를 곱하고 c를 더한 값과 같다는 것은 잘 알텐데, e(t)항은 white noise라고 부르며 평균이 0 이고 분산이 1인 정규분포에서 도출된 random값을 이야기 한다. 즉 X(t)값은 평균이 a * X(t - 1) + c이며 분산이 u인 정규 분포에서 도출되는 임의의 값이다. 만약 더 이전의 시점(P)를 모델에 합치고 싶다면 AR(P)모형이 되는 것이다.

$$X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t^* .$$

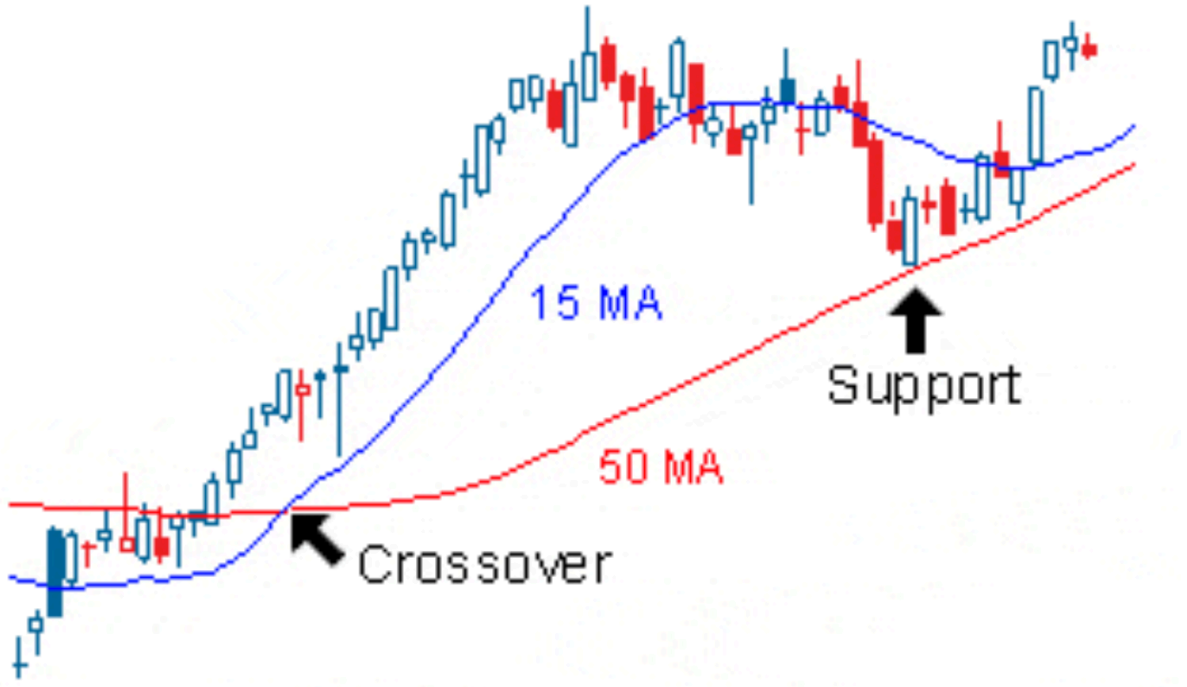
2. MA (MovingAverage) model



시간이 지날수록 어떠한 Random Variable의 평균값이 지속적으로 증가하거나 감소하는 경향을 보일 수 있다. 예를 들어 봄에서 여름이 될수록 일반적으로 가계 전기 소요량은 증가하고, 여름에서 가을이 될수록 내려가다가, 다시 겨울이 될수록 증가하는 경향이 있다는 것을 우리 모두 잘 알고 있을 것이다. 전월의 전기세가 돌아오는 월의 전기세에 영향을 끼치지 않는 다고 가정 할 수 있고, 전기 사용량이 얼마라고 정확히 이야기 할 수 없기에 평균이동이 있는 시계열 데이터가 될 것이라고 예측할 수 있다. 데이터의 평균값 자체가 시간에 따라 변화하는 경향이 바로 Moving Average 모형이다. 이동평균을 시계열 모형으로 구성한 것을 MA 모형이라고 부르는데, 간단한 형태가 직전 데이터가 다음 데이터에 영향을 준다고 가정하는 MA(1)모형이다.

X(t) = a * e(t - 1) + c + u * e(t)

시점 t에서 얻게 될 X(t)의 평균값은 시점 t - 1에서 발생한 error에 a를 곱하고 c를 더한 값과 같다는 뜻이다. 즉, X(t)값은 평균이 a * e(t - 1) + c 이며 분산이 u인 정규분포의 임의의 값이다. AR(1)모형과 가장 결정적인 차이는 이전에 발생한 error가 중요하지, 이전에 발생한 데이터의 값, 즉 X(t - 1)이 무엇 인지는 중요치 않다는 것이다. 만약 더 이전의 시점 (P시점)을 모델에 넣고자 하면 MA(P)모형이 되는 것이다.



예를 들어서 50일 평균값보다 최근 15일 평균값의 이동폭이 더 크다면 주가가 치솟는다. 즉 골든 크로스가 발생한다고 볼 수 있다.

3. ARMA (AutoRegressive Moving Average) model

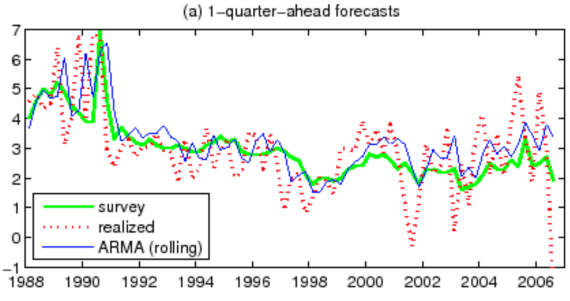
AR모델과 MA모델을 합치는 것을 의미한다. 연구기관에서 주로 사용하며 보통 ARMA(2,2)가 가장 선호되는 모델이다. 가장 단순한 형태는 ARMA(1, 1)모형은 다음과 같다.

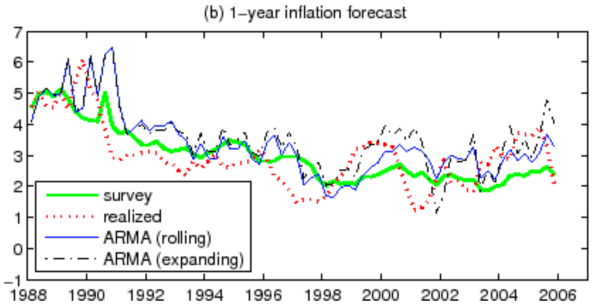
X(t) = a * X(t - 1) + b * e(t-1) + c + u * e(t)

평균이 a * X(t - 1) + b * e(t-1) + c 이고, 분산이 u인 임의의 데이터다. 전의 데이터 값도 영향을 받고, 전의 error수치에도 영향을 받는 상황을 의미한다.

ARMA(2,2)

X(t) = a1 * X(t-1) + a2 * X(t-2) + b1 * e(t-1) + b2 * e(t-2) + c + u * e(t)





2014147513 조문기

여론조사결과(X)와 실제 발생량의 차이(e)가 가지는 패턴을 이용해서 보다 나은 추정을 할 수 있지 않을까 하는 시도가 가능하다. 이 이후 내삽과 외삽이 가능하다.

4. ARIMA (AutoRegressive Integrated Moving Average) model
   ARMA모형이 과거의 데이터들만 사용하는 것이 비해섯 ARIMA모형은 이것에 더해 과거의 데이터가 지니고 있던 추세(momentum)까지 반영하게 된다. 즉 Correlation뿐 아니라 Cointegration까지 고려한 모델이다. Co-integration은 Cor-relation이 선형관계를 설명한다면 추세관계를 설명한다.

   선형관계
   두 변수 X-Y간에 Correlation이 0보다 크면 X가 큰 값이 나올때 Y도 큰 값을 가진다.
   두 변수 X-Y간에 Correlation이 0보다 작으면 X가 큰 값이 나올때 Y는 작은 값을 가진다.

   추세관계
   두 변수 X-Y간에 Cointgration이 0보다 크면 X의 값이 이전 값보다 증가하면 Y의 값도 증가한다.
   두 변수 X-Y간에 Cointegration이 0보다 작으면 X의 값이 이전 값보다 증가하면 Y의 값은 감소한다.

   + EX)
   correlation > 0 && cointegration < 0
   X가 큰값이며 증가하는 추세일 경우 Y는 현재 큰 값이나 빠르게 감소하는 추세로 반응.
   그러나 이 모델은 white noise의 추세는 고려하지 않는다. white noise는 추세가 존재하면 안되기 때문이다. 이런 이유 때문에 사실은 ARMA모델과 다를 바가 없다. 결국은 white noise가 어떻게 결정되느냐에 따라서 식이 변하게 되기 때문이다. 많은 학자들은 ARIMA모델보다 ARMA모델을 선택해서 사용한다. ‘추세’라는 것이 정당한 데이터라고 생각 할 수 없는 상황이 존재하기 때문이다.

RCNN의 도입 배경

- DCNN (deep convolution neural network)는 컴퓨터 비전 문제에서 대단하게 작용한다.
- 실제 세상에서 글자 / 손글씨 / 악보 인식 등은 사실 서로 서로가 독립적인 sequence가 아니라 시간에 따른 영향을 받는다는 것을 인지하게 되었다.
- 일반적 데이터 인식이 아니라, make_pair(data, time)으로 이루어진 시계열 데이터 인식 문제로 가게 되었다. 또한 시계열 데이터의 특징은, 길이에 구애받지 않는다는 것이다. OK (2) / CONGRATULATIONS (15) 등 그 길이들이 다르다.

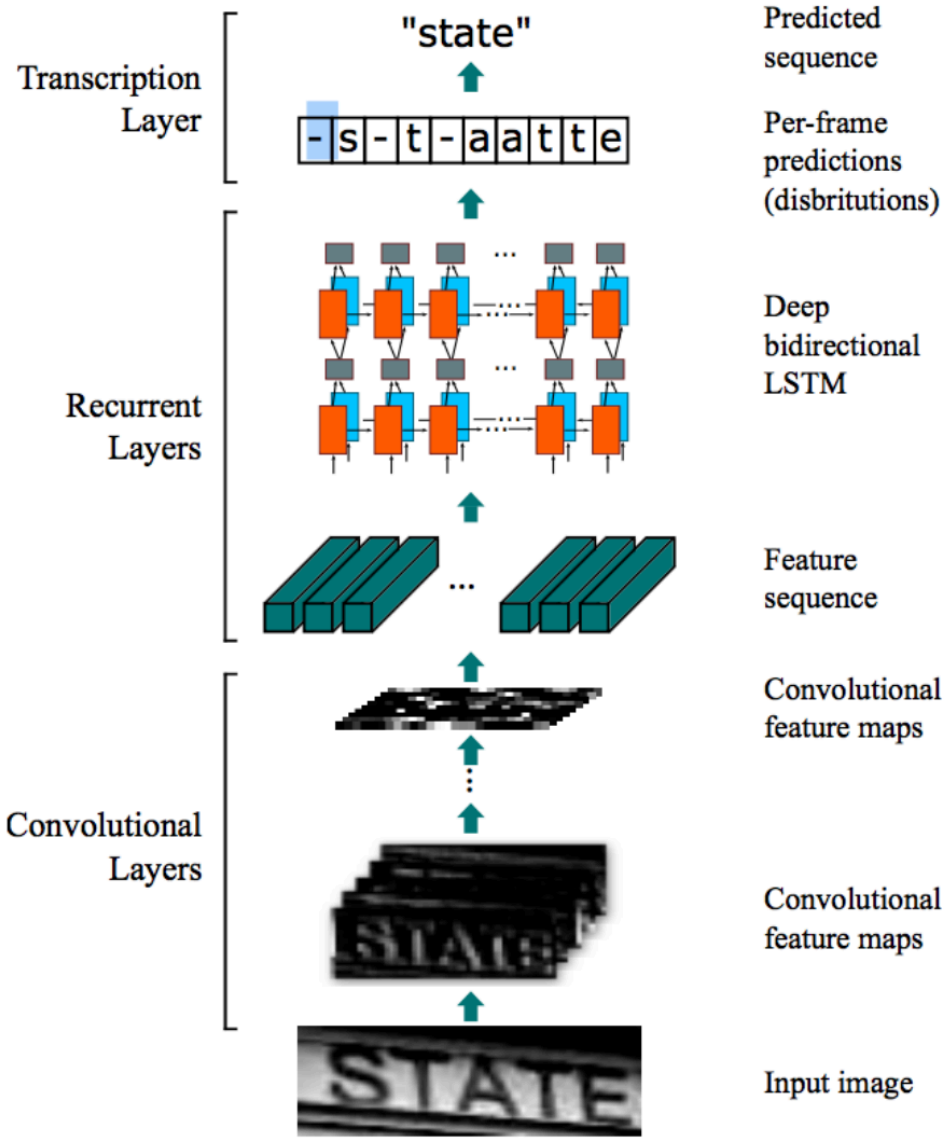
2014147513 조문기

- 결론적으로 말하자면, DCNN은 이러한 가변적인 상황에 대해서 대응 할 수 없다는 것이다. (보통 고정된 입출력 차원으로 작동하고 다양한 label sequence제작이 불가능)
- DCNN차원에서 이를 해결하기 위해서 글자 하나하나를 발견하고 이 발견된 글자들을 토대로 DCNN모델로 인식하려 했다. (label 된 글자 이미지로 학습) 이 방법은 정확한 발견과 원래 글자 이미지에서 각각을 cropping하는 강력한 글자 detector의 학습을 필요로 했다. 즉 다음과 같은 예시인 것이다.
- CONGRATULATIONS -> 각각의 영어 단어를 이미지 분류 문제로 보고 각각 문자들을 label 함(매우 큰 숫자의 class를 학습 모델로 필요로 함) -> 그후 이 인식된 문자들을 토대로 단어를 인식하게 함.
- DROWBACKS : 결국은 단어의 문자들을 분류를 해야하는데, 중국어(기본 조합이 100만이 넘어감) 나 악보와 같은 상황은 처리되기 어려운 문제로 다가옴.
- 결론 : DCNN에 기초한 현재 시스템들은 이미지 기반 시계열 인식에 바로 적용 불가능.
- DNN의 한 중요한 분야인 RNN에서 장점은 학습과 테스트에서 시계열 이미지들의 position 이 필요 없다는 것임.
- 그러나 입력 이미지를 이미지 feature의 sequence로 변환하는 전처리 단계가 필수적이다.
- 예를 들자면 다음과 같은 상황을 필수적으로 요구한다는 것
  1. Graves는 손글씨에서 기하학적 구조 및 이미지 특징을 추출했다.
  2. Su & Lu는 단어 이미지를 시계열 HOG(Histogram of Oriented Gradients) feature로 변환했다.
- 전처리 단계는 파이프 라인의 후속 구성 요소와 독립적이므로 end-to-end학습을 할 수 없었다. (전처리는 RNN construction과는 관련이 없기 때문)
- 신경망 구조를 기반으로 하는 기존의 방식에 더불어 새로운 기술들도 생기게 되었다.
- Almaza `n과 Rodrigues-Serrano는 단어 이미지와 텍스트 문자열을 일반적인 벡터 공간의 부분 공간에 포함시키고 단어 인식을 검색 문제로 변환하는 방법을 제안했습니다.
- Yao와 Gordo는 글자 인식을 위해 중간 단계의 특징을 사용했습니다.

CRNN 특유의 장점

1. 상세한 주석(EX : 글자) 이 필요없는 시계열(EX : 단어) 을 학습 할 수 있다.
2. 이미지 데이터에서 직접 정보를 얻는 representation을 배우는데 동일한 속성을 지닌다. (사람이 만든 특징이나 전처리 방법{이진화, 세분화, 구성 요소 지역화}가 필요하지 않음)
3. RNN과 동일한 속성을 지니며 시계열 label을 생성 할 수 있다.
4. 학습과 테스트 단계 모두에서 height normalization(이미지 크기 정규화)만 필요로 하는 시계열 데이터의 길이에 제약을 받지 않는다.

CRNN의 구조



2014147513 조문기

- convolution Network의 최상부에서 Convolution layer에 의해 출력된 feature sequence의 각 프레임에 대한 예측을 하기 위해서 RNN이 구축됩니다.
- CRNN은 결국 Convolution layer + Recurrent layer + Transcription layer로 구성된다.
- CRNN의 하단에서 Convolution layer는 각 입력 이미지에서 feature sequence를 자동으로 출한다.
- 상단의 Transcription layer는 Recurrent lay에 의한 프레임 별 예측 데이터를 label sequence로 변환하는데 사용한다.
- CRNN은 여러 종류의 Network 구조 (cf. CNN & RNN)로 구성되지만 하나의 loss function으로 공동으로 교육 할 수 있다.

---

loss function?!

- 손실 함수란 신경망이 학습할 수 있도록 해주는 지표이다. 머신러닝 모델의 출력값과 실험자가 원하는 출력값의 차이, 즉 오차를 말한다. 이 손실 함수 값이 최소화되도록 하는 가중치(weights)들을 구하는게 목표인 것이다. 손실함수는 크게 2가지로 나누어 진다.

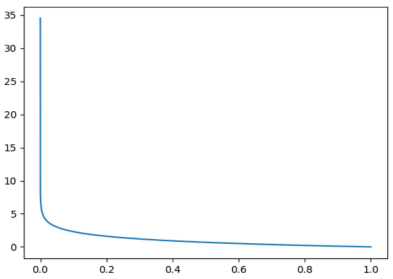
1. 평균 제곱 오차 (MSE)

$$E = \frac{1}{2}\sum_{k}(y_k - t_k)^2$$

평균 제곱 오차 (Mean Squared Error)는 계산이 간편하여 가장 많이 사용되는 손실 함수이다. 각 출력값에 해당하는 것과 실제 원하는 출력값의 차이들의 거리들의 제곱을 더한다고 생각하면 된다. 실제로는 1 / 2가 아니라 1 / N이라고 생각하면 편하다. 거리 차이를 제곱하는 것의 추가적 장점으로는 거리 차이가 작은 데이터는 계속 작아지고, 거리 차이가 큰 데이터는 더 커지는 상황이 놓여 어느 부분이 오차를 가장 극단적으로 만드는지 찾기 쉬워진다.

2. 교차 엔트로피 오차 (CEE)

$$E = -\sum_{k} t_k \log y_k$$



교차 엔트로피 오차 (Cross Entropy Error)는 기본적으로 분류(Classification)문제에서 one-hot-encoding을 해야지만 사용할 수 있는 오차 계산법이다. 저번 시간의 RNN에서 h e l o -> 1,0,0,0 / 0,1,0,0 / 0,0,1,0 / 0,0,0,1과 같이 나눈 방식이 one-hot-encoding method다. 식의 t값이 각각의 벡터이고, 거기에다가 모델의 출력 값에 자연로그를 취한 것(logy)이 곱해지는 형태이다. 분류 문제에서 모델의 출력은 0 ~ 1이므로 -log(y)에 대해서 생각해보자면, 정답이 1이라고 했을 경우 오차가 0에 수렴한다. 그러나 0에 가까워지면 기하급수적으로 증가하는 것을 알 수 있다. 즉 정답에 멀어질수록 큰 패널티를 부여하는 것이다. 이 이유는 일반적으로 분류 문제에서는 데이터의 출력을 0과 1로 구분하기 위해서 sigmoid함수를 사용한다. 시그모이드 함수 식에는 자연상수 e가 포함되기 때문에 기존의 평균 제곱 오차를 사용하면 매끄럽

지 못한 그래프의 개형이 생긴다. 이때 만약 경사 하강법(Gradient Descent Algorithm)을 사용하면 전역 최소점(global minimum)을 찾지 못하고 지역 최소점(local minimum)에 걸릴 가능성이 크다. 그래서 자연 상수 e에 반대되는 자연 로그를 모델의 출력 값에 취하는 손실함수를 사용하는 것이다.

- 손실함수의 목적은 머신러닝 모델의 최종 목적과 같다. 높은 정확도를 끌어내는 매개변수 _parameter(가중치, 편향)을 찾는 것이다. 최적의 매개변수를 탐색할때 손실함수의 값을 가능한 한 작게 하는 매개변수 값을 찾는다. 이때 매개변수의 기울기를 계산하고(미분), 그 미분 값을 토대로 매개변수 값을 갱신하는 과정을 반복한다.
- 중요한 점은, 정확도와는 달리 손실 함수는 매개변수의 변화에 따라 연속적으로, 즉 같은 배를 타고 있다는 점이다. 손실 함수와는 달리 정확도는 매개변수의 변화에 둔감하고, 변화가 있다고 해도 불연속적으로 작동한다. 그렇기에 정확도가 아닌 손실함수를 지표로 삼아 ML을 진행하는 것이다.

Sigmoid function

$$S(t) = \frac{1}{1 + e^{-t}}$$

- sigmoid function의 사용 이유는 블럭형식의 계단형 함수(미분이 불가능함)를 미분이 가능한 함수로 부드럽게 구현하기 위함이다.
- e^t형태의 값을 제시해서 출력을 제한한다. 출력값 0 ~ 1 사이의 값 하나이다. 그래서 이를 통해서 역치를 함께 사용하면 T / F를 결정지을 수 있다.

Softmax function

- softmax function의 사용 이유는 입력받은 값을 출력으로 0 ~ 1 사이의 값으로 모두 정규화하며 그 출력 값들의 총합은 항상 1이 되는 특성을 가지고 있다.
- 분류하고 싶은 클래스의 수 만큼 출력으로 구성한다. 그중 가장 큰 출력값의 클래스가 확률이 가장 높은 것으로 간주된다.
- 그러나 만약에 softmax function에 의해서 나온 결과값이 [0.4, 0.3, 0.2, 0.1]로 나와서 1등한 0.4와 [0.7, 0.1, 0.1, 0.1]으로 나와 1등한 0.7은 그 결정에 대한 확신이 다르기 때문에 추가 판단하기도 한다.
- 소프트맥스를 통해 구하는 식은 다음과 같다.

$$f(\vec{x})_i = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}} \quad \text{for } i = 1, \ldots, K$$

- 입력값의 대소 순서가 출력값의 대소 순서와 같다는 것을 알 수 있
다. 즉 가장 큰값은 이미 소프트맥스 이전에도 가장 큰 값이라고 예측이 가능하다. 그래서 추론(운영)단계에서 연산속도를 빠르게 하기 위해서 생략하기도 한다.
- 결과값을 One-Hot-Encoding의 입력으로 연결하면 가장 큰 값만 True로 이용 가능하다.

2014147513 조문기

- 만약 입력값을 [1.0, 1.0, 2.0]을 넣으면 소프트맥스 함수식에 의해서 [0.25, 0.25, 0.5]가 나오지 않고 [0.2, 0.2, 0.6]이 나오게 되는데, 이 이유는 exp(x)의 영향이다. 소프트맥수 함수는 1.0과 2.0을 계산하는 것이 아니라 e^1 = 2.7, e^2 = 7.3값을 계산에 이용하기 때문에 이 비율이 2배가 아니라 3배정도로 계산이 되는 것이다. 즉 입력값이 커짐에 따라서 기울기가 증가하게 된다.

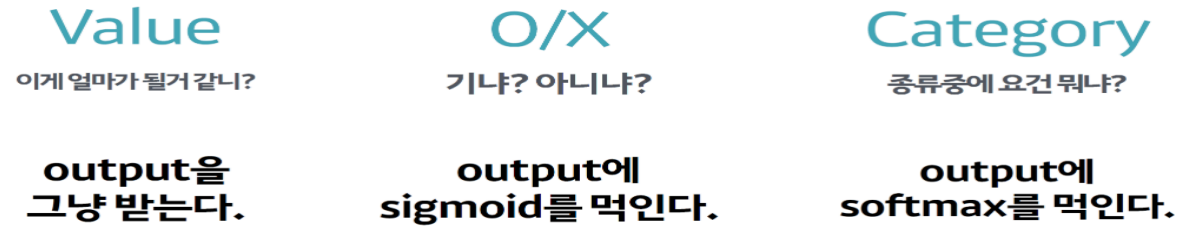
CE (Cross Entropy) 계산 방식

- 앞서 CEE에서 이야기한, 음의 로그우도를 계산하는 방식에 대해서 추가적으로 이야기를 하자면, 0 * log0의 값이 나오는 경우가 있다. 이 경우는 0으로 취급을 하게 된다.

$$H(P, Q) = -\sum_{x} P(x) \log Q(x)$$

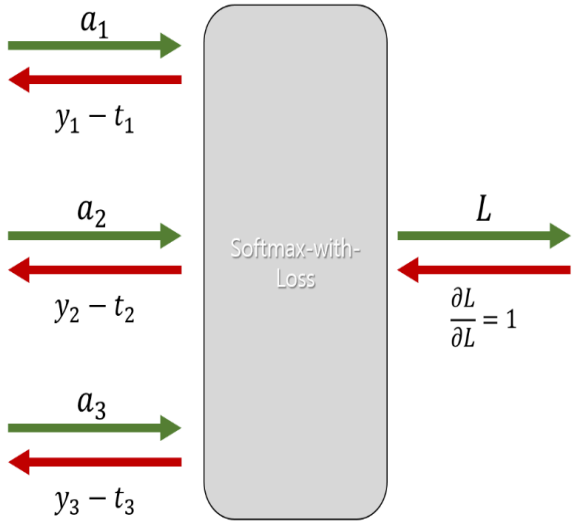
- 예를 들어서 범주가 2개이고 정답 레이블이 [1, 0]인 관측치 x가 있다고 가정을 하면, P는 우리가 가지고 있는 데이터의 분포를 나타내기 때문에 첫번째 범주일 확률이 1, 두번째 범주일 확률이 0이다. 우리가 학습시킨 Q가 P에 근접하게 하고 싶은 것이 우리의 목표인데, 만약 Q의 학습이 이상하게 되어서 결과값이 [0, 1]로 나왔다고 하면 계산식은 다음과 같다.
- - [1 0][log0 log1]^T = -(1log0 + 0log1) = -(-∞) = ∞
- 학습이 잘 되어서 [1, 0]의 Q가 나온 경우는
  - [1 0][log1 log0]^T = -(1log1 + 0log0) = -(0) = 0
- 그래서 loss는 적은 쪽인 두번째 선택을 하도록 돕게 되는 것이다. 그리고 MSE보다 CEE가 우수한 성능을 보이게 되는데, 그 이유는 다음과 같다.

1. 동일한 기능
   우리가 만들고 싶은 모델을 가우시안 분포로 가정을 한다면, 결국 크로스 엔트로피 최소화 방법과 평균제곱오차 최소화 방법은 본질적으로 동일하다. 크로스 엔트로피 최소화 방법은 현재 가지고 있는 데이터의 분포와 모델의 가우시안 분포 간의 간극을 최소화한다는 의미이기 때문이다.
   추가적으로 만약 모델이 베르누이 분포로 가정을 하면, 우리가 가진 데이터의 분포와 모델의 베르누이 분포 간 차이가 최소화 하는 방향으로 학습이 이루어 진다.
   loss function을 사용할때 결국 MSE보다는 CEE의 선택이 더 높은 효율성을 가진다.



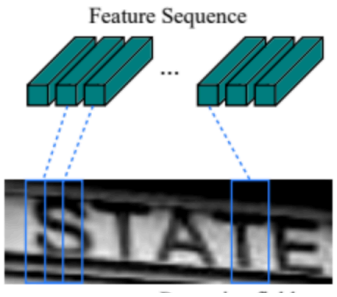
2014147513 조문기

2. 역전파(back-propagation)의 gradient dying 방지
   크로스 엔트로피를 사용하면 딥러닝 역전파에서 값이 0으로 넘어가는 것을 어느정도 막을 수 있다. 또한 구하는 과정도 계산적 가중치가 그리 크지 않다.
   예를 들어 우리가 구축한 모델이 다항분포고, 최종 output node는 3차원 vector를 받는 softmax, loss는 CE일 경우를 그림으로 도식화 하면 다음과 같다.



softmax_with_loss 노드는 a들을 입력으로 받은 후 softmax를 통해 확률값으로 전환한다. 그 후 이를 바탕으로 loss function을 사용해 L이라는 loss를 산출한다. 반대의 역전파하는 그래디언트는 yk - tk가 된다. 만약 진짜 정답이 t3이라면, back_propagation의 gradient는 y1, y2, y3 - 1로 구현이 가능하다. 이뿐 아니라 정답이 아닌 노드의 손실에 대한 그래디언트는 소프트맥스 확률값이고, 정답 레이블에 해당하는 노드의 그래디언트는 여기에서 1을 빼준 값이기 때문에 0으로 되는 경우는 많지 않다.

CRNN의 작동 방식



1. pre_for CNN
   STATE라는 이미지를 알아서 S, T, A, T, E라고 이미지의 특정 부분을 알아서 인식 할 수 없기 때문에 이미지를 각각의 feature sequence로 나눈다.

   1. Prerequisite for train model
      그래서 CNN을 적용하기 이전에 prerequisite로 feature sequence의 길이 (이미지의 width와 CNN의 stride)를 고려하여 변환해야한다. (그래서 size에 따라서 S가 하나의 layer로 변할 수도 있고, ST가 하나의 layer로 변할 수도 있고, S의 절반이 하나의 layer로 변할 수도 있는 것이다)

   2. Prerequisite for inference
      여러 텍스트 sequence가 포함된 이미지일 경우 서로가 섞이면 곤란한 상황이 일어나기 때문에 하나의 텍스트 시퀀스만 포함하도록 추출하는 작업이 필요하다.

2. CNN
   1-1과 1-2를 통해서 텍스트 시퀀스가 있는 feature Sequence만 사용하도록 바로 사용되는 것은 아니다. prerequisition도 결국은 loss function을 통해서 계속 조정하게 된다. 만약 학습이 잘 되어서 적합한 사전 준비가 끝났다면 이미지의 합당한 크기를 통해서 생긴 layer

들을 convolution한 값이 나올 것이다. 이 값들은 이제 각각의 input sequence로 RNN에 입력하게 된다.

3. RNN
   RNN의 경우 LSTM을 주로 사용한다. layer들이 단일 차원이 아니라 다차원으로 입력이 가능하다. 그래서 network layer가 3차 4차 … 쌓일때 인풋 레이어가 1차원이 아니기 때문에 많은 계산을 요구한다. loss계산은 보통 RNN에 포함되어 있다.

   1. Bidirectional LSTM vs LSTM
      RNN정리글에서 추가해서, 단방향 LSTM과 양방향 LSTM의 차이점에 대해서 이야기하고자 한다. LSTM은 핵심에 숨겨진 상태를 사용해 이미 통과 한 입력의 정보를 보존한다. 그런데 단방향 LSTM은 과거의 정보만 보존한다. 이에 비해서 양방향을 사용하면 입력 내용이 과거에서 미래로, 미래에서 과거로 두가지 방식으로 작동한다. 만약에 다음과 같은 문장이 있는데 중간의 단어가 없는 상황이라고 가정하자.
      그 애들은 [ ]에 갔다.
      단방향 LSTM의 경우 “그 애들은” 만 가지고 [ ]를 예측하려고 할 것이다.
      그러나 양방향 LSTM의 상황에서는
      “그 애들은 [ ]에 갔다. 그리고 1시간 후 그들은 수영장에서 나왔다.” 라는 문장을 통해서 빈 공간의 단어를 예측하게 된다. 그래서 미래의 정보를 사용해 네트워크가 다음 단어가 무엇인지 쉽게 이해할 수 있다는 것이 원리이다.

   2. BiLSTM구현 방법
      1) BiLSTM구현은 LSTM모듈을 2개를 병합하여 양방향을 만드는 방법이 있다.
      2) Bidirectional이라는 클래스가 생겨서 이를 이용해 추가가 가능하다. 활성화 함수도 적용 가능하다.

4. Transcription Layer
   Transcription은 RNN으로부터 만들어진 프레임당 예측을 라벨 시퀀스로 변환화는 과정이다. 수학적으로는 프레임당 예측이 가장 높을 확률과 함께 라벨 시퀀스를 찾는 것이다. 위 사진에서 [__SS___TA___TTEE]와 같은 상황이 나올때, 2가지 방법을 통해서 처리하게 된다.

   1. Lexicon-free
      사전이 존재하지 않는 상황에서 확률을 통해서 sequence를 결정짓게 된다.

   2. Lexicon-based
      사전이 존재하는 상황인 lexicon-based mode에서는 예측과정을 사전을 통해서 처리하게 된다. lexicon file을 구해서 처리하게 된다. vaderSentiment와 같은 파일들은 단어에 대한 감성을 가지고 있는 파일도 있고 많은 lexicon file들이 존재한다.