

December 10, 2025

- 강의명: CSCI E-89B: 자연어 처리 입문
- 주차: Lecture 02
- 교수명: Dmitry Kurochkin
- 목적: Lecture 02의 핵심 개념 학습

Abstract

본 문서는 텍스트 데이터를 수치로 변환하는 TF-IDF 기법부터 시작하여, 순차적 데이터 처리에 특화된 순환 신경망(RNN)의 기초와 한계, 그리고 이를 극복하기 위한 LSTM, GRU와 같은 고급 모델까지의 핵심 내용을 통합하여 정리합니다. 각 개념은 구체적인 계산 예시와 응용 사례를 통해 설명되며, 신경망 학습 과정에서 발생하는 주요 문제들과 해결책을 다룹니다. 이 자료는 단독으로 학습이 가능하도록 구성되었습니다.

Contents

I 텍스트 정량화와 분류	2
1 개요	2
2 TF-IDF를 이용한 텍스트 표현	2
2.1 TF-IDF의 개념	2
2.2 TF-IDF 계산 방법	2
3 응용 사례: 특허 데이터 분류	4
3.1 프로젝트 목표 및 데이터 준비	4
3.2 차원 축소: t-검정(t-test)의 활용	4
3.2.1 t-검정 기반 피처 선택 절차	4
II 신경망 기초와 순환 신경망(RNN)	6
4 신경망의 기본 연산	6
4.1 순전파 (Forward Propagation)	6
4.2 역전파 (Backward Propagation)	6

4.3	가중치와 하이퍼파라미터	7
5	신경망 학습의 핵심: 경사 하강법	8
5.1	학습률(α)의 중요성	8
III	순차 데이터 처리를 위한 순환 신경망(RNN)	9
6	순환 신경망(RNN)의 등장 배경	9
7	기본 RNN (Vanilla RNN) 구조	9
7.1	RNN의 작동 원리	9
7.2	RNN의 파라미터 수 계산	10
8	RNN의 한계: 기울기 문제	11
8.1	기울기 소실 (Vanishing Gradient)	11
8.2	기울기 폭주 (Exploding Gradient)	11
IV	장기 의존성 문제 해결을 위한 고급 RNN 모델	12
9	LSTM (Long Short-Term Memory)	12
9.1	LSTM의 주요 구성 요소	12
10	GRU (Gated Recurrent Unit)	12
11	양방향 RNN (Bidirectional RNN)	14
11.1	Bi-RNN의 구조와 원리	14
V	FAQ 및 요약	15
12	자주 묻는 질문 (FAQ)	15
13	한 페이지 요약	15

Part I

텍스트 정량화와 분류

1 개요

자연어 처리의 첫 단계는 컴퓨터가 이해할 수 있도록 텍스트를 숫자 형태의 벡터로 변환하는 것입니다. 이 과정에서 가장 널리 사용되는 기법 중 하나가 **TF-IDF**입니다.

TF-IDF는 각 문서에서 특정 단어가 얼마나 중요한지를 나타내는 통계적 수치입니다. 이 수치를 이용해 텍스트의 내용을 정량화하고, 이를 바탕으로 문서 분류와 같은 머신러닝 작업을 수행할 수 있습니다.

본 파트에서는 TF-IDF의 원리를 계산 예시와 함께 살펴보고, 실제 특허 문서를 산업 분야별로 분류하는 응용 사례를 통해 데이터 준비부터 모델 학습까지의 전 과정을 학습합니다. 특히, 수많은 단어 중 분류에 실질적으로 도움이 되는 핵심 단어를 선별하기 위해 통계적 기법인 **t-검정(t-test)**을 활용하는 차원 축소 방법을 심도 있게 다룹니다.

2 TF-IDF를 이용한 텍스트 표현

2.1 TF-IDF의 개념

TF-IDF는 **Term Frequency**(단어 빈도)와 **Inverse Document Frequency**(역문서 빈도)라는 두 가지 값을 곱하여 계산됩니다.

- **TF (Term Frequency, 단어 빈도)**: 특정 문서 내에서 한 단어가 얼마나 자주 등장하는지를 나타냅니다. 자주 나올수록 해당 문서 내에서 중요할 가능성이 높습니다.
- **IDF (Inverse Document Frequency, 역문서 빈도)**: 전체 문서 집합에서 특정 단어가 얼마나 희귀한지를 나타냅니다. 여러 문서에 공통으로 등장하는 단어(예: a, the, and)는 중요도가 낮은 반면, 소수의 문서에만 나타나는 단어는 해당 문서의 주제를 잘 나타내므로 중요도가 높습니다.

TF-IDF 핵심 원리

한 문서 안에서는 자주 등장하지만(**TF가 높음**), 전체 문서들 중에서는 드물게 나타나는 단어(**IDF가 높음**)가 해당 문서를 대표하는 중요한 단어라는 아이디어에 기반합니다.

2.2 TF-IDF 계산 방법

간단한 예시를 통해 TF-IDF 계산 과정을 단계별로 살펴보겠습니다. 3개의 문서가 있고, 첫 번째 문서에서 'cat'이라는 단어의 TF-IDF를 계산해 봅시다.

- 문서 1: "the cat set on the mat" (총 6개 단어)
- 문서 2: "the cat did something again"
- 문서 3: "some sentence but no word"

1. **TF(cat, 문서 1) 계산**: 문서 1에서 단어 'cat'의 등장 횟수는 1번이고, 전체 단어 수는 6개입니다.

$$\text{TF}(\text{'cat'}, \text{문서 1}) = \frac{\text{'cat'의 등장 횟수}}{\text{문서 1의 전체 단어 수}} = \frac{1}{6}$$

2. **IDF(cat) 계산:** 전체 3개의 문서 중 'cat'이 포함된 문서는 2개입니다. IDF는 이 비율에 역수를 취해 계산합니다.

$$\text{IDF('cat')} = \log \left(\frac{\text{전체 문서 수}}{\text{'cat'을 포함한 문서 수} + 1} \right)$$

여기서 분모에 1을 더하는 것은 특정 단어가 모든 문서에 등장하여 분모가 0이 되는 것을 방지하는 스무딩(smoothing) 기법입니다. 로그를 취하는 것은 값의 스케일을 조절하기 위함입니다.

$$\text{IDF('cat')} = \log \left(\frac{3}{2+1} \right) \text{ 또는 간단히 } \log \left(\frac{3}{2} \right)$$

계산 방식에는 여러 변형이 존재하며, 라이브러리마다 기본값이 다를 수 있습니다.

3. **TF-IDF(cat, 문서 1) 계산:** TF와 IDF 값을 곱하여 최종 점수를 얻습니다.

$$\text{TF-IDF('cat', 문서 1)} = \text{TF('cat', 문서 1)} \times \text{IDF('cat')} = \frac{1}{6} \times \log \left(\frac{3}{2} \right)$$

실무적 고려사항

정규화(Normalization): 실제 라이브러리에서는 계산된 TF-IDF 벡터의 크기를 1로 만드는 정규화 과정을 거칩니다. 이는 문서 길이에 따른 편향을 줄여줍니다.

학습/테스트 데이터 분리: 모델 학습 시, IDF 값은 **학습 데이터(train data)**만으로 계산해야 합니다. 테스트 데이터에 적용할 때는 이 학습된 IDF 값을 그대로 가져와 사용합니다. 이는 테스트 데이터 정보가 모델 학습에 미리 유출되는 것을 막기 위함입니다.

3 응용 사례: 특허 데이터 분류

3.1 프로젝트 목표 및 데이터 준비

이 프로젝트의 목표는 1895년부터 1935년 사이의 미국 특허 문서를 자동차 산업 관련 특허와 비관련 특허로 자동 분류하는 것입니다.

데이터 소스 특허 제목(title)과 본문(description) 텍스트 데이터.

학습 데이터 생성

- **레이블 1 (자동차 관련):** 당시 자동차를 생산했던 기업 목록을 확보하여, 해당 기업들이 출원한 특허를 모두 자동차 관련으로 간주하고 레이블 '1'을 부여합니다.

- **레이블 0 (비관련):** 전체 특허 풀에서 무작위로 샘플을 추출하여 레이블 '0'을 부여합니다. 당시 수많은 분야의 특허가 존재했으므로, 무작위 샘플은 대부분 자동차와 무관할 것이라는 가정에 기반합니다.

데이터 레이블링의 한계

이 방식은 100% 정확하지 않습니다. 자동차 회사가 아닌 독립 연구자가 출원한 관련 특허는 '0'으로 잘못 분류될 수 있고, 무작위 샘플에 우연히 자동차 관련 특허가 포함될 수도 있습니다. 하지만 대규모 데이터를 다룰 때 현실적인 접근 방식입니다.

3.2 차원 축소: t-검정(t-test)의 활용

특허 문서에는 수만 개 이상의 고유한 단어가 존재합니다. 이 모든 단어를 모델의 입력 피처(feature)로 사용하면 계산량이 방대해지고, 모델 성능이 저하되는 **차원의 저주(curse of dimensionality)** 문제가 발생합니다.

따라서 분류에 실질적으로 도움이 되는 핵심 단어들만 선별하는 과정이 필요합니다. 이때 **t-검정(t-test)**을 활용할 수 있습니다.

[title=t-검정(t-test)] 두 집단의 평균값에 통계적으로 유의미한 차이가 있는지를 검정하는 방법입니다. 여기서는 특정 단어('engine' 등)의 TF-IDF 점수가 '자동차 관련 특허 집단(레이블 1)'과 '비관련 특허 집단(레이블 0)' 사이에서 평균적으로 차이가 나는지를 확인합니다.

3.2.1 t-검정 기반 피처 선택 절차

1. **단어별 t-검정 수행:** 모든 고유 단어에 대해 다음을 수행합니다.
 - 집단 1: 자동차 관련 특허들에서 해당 단어의 TF-IDF 점수 분포
 - 집단 2: 비관련 특허들에서 해당 단어의 TF-IDF 점수 분포
 - 두 집단의 평균 TF-IDF 점수 차이에 대한 t-검정을 실시합니다.
2. **p-값(p-value) 확인:** t-검정 결과로 p-값을 얻습니다.
 - **p-값이 작다:** 두 집단 간 평균 차이가 우연히 발생했을 확률이 낮다는 의미입니다. 즉, 해당 단어는 두 집단을 구분하는 데 **매우 유의미**합니다. (예: 'engine', 'wheel')
 - **p-값이 크다:** 두 집단 간 평균 차이가 뚜렷하지 않다는 의미입니다. 해당 단어는 분류에 별 도움이 되지 않습니다. (예: 'the', 'is', 'and')
3. **핵심 단어 선택:** 모든 단어의 p-값을 오름차순으로 정렬한 뒤, p-값이 가장 작은 상위 N개(예: 300개)의 단어만 최종 피처로 선택합니다.

t-검정을 이용한 차원 축소의 효과

t-검정을 통해 'and', 'of'와 같이 분류에 기여하지 못하는 일반적인 단어들은 제거하고, 'engine', 'chassis', 'vehicle'처럼 특정 도메인을 강력하게 나타내는 단어들만 선별할 수 있습니다. 이를 통해 모델의 성능과 효율을 크게 향상시킬 수 있습니다.

Part II

신경망 기초와 순환 신경망(RNN)

4 신경망의 기본 연산

신경망은 입력 데이터로부터 복잡한 패턴을 학습하여 예측 결과를 출력하는 모델입니다. 이 과정은 크게 순전파(Forward Propagation)와 역전파(Backward Propagation) 두 단계로 이루어집니다.

4.1 순전파 (Forward Propagation)

순전파는 입력 데이터가 신경망의 각 층(layer)을 순서대로 거쳐 최종 출력값(\hat{y})을 계산하는 과정입니다. 각 뉴런은 이전 층의 출력에 가중치(w)를 곱하고 편향(b)을 더한 뒤, 활성화 함수(f)를 적용하여 다음 층으로 신호를 전달합니다.

□ 예제: title

입력 $x_1 = 1.3, x_2 = 0.7$ 이고, 활성화 함수가 $\text{ReLU}(f(z) = \max(0, z))$ 일 때의 계산 과정입니다.

1. 첫 번째 은닉층(Hidden Layer) 계산

- 첫 번째 뉴런(u_1):

$$\begin{aligned} z_1 &= w_{01}^{(1)} + w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 \\ &= -1.2 + (0.1 \times 1.3) + (0.5 \times 0.7) = -0.72 \\ u_1 &= f(z_1) = \text{ReLU}(-0.72) = 0 \end{aligned}$$

- 두 번째 뉴런(u_2):

$$\begin{aligned} z_2 &= w_{02}^{(1)} + w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 \\ &= 0.9 + (0.8 \times 1.3) + (0.3 \times 0.7) = 2.15 \\ u_2 &= f(z_2) = \text{ReLU}(2.15) = 2.15 \end{aligned}$$

2. 출력층(Output Layer) 계산

- 최종 출력(\hat{y}):

$$\begin{aligned} z_{out} &= w_0^{(2)} + w_1^{(2)} u_1 + w_2^{(2)} u_2 \\ &= 0.2 + (0.8 \times 0) + (1.2 \times 2.15) = 2.78 \\ \hat{y} &= f(z_{out}) = \text{ReLU}(2.78) = 2.78 \end{aligned}$$

이처럼 입력에서 출력 방향으로 차례대로 값을 계산해 나가는 것이 순전파입니다.

4.2 역전파 (Backward Propagation)

역전파는 신경망의 예측값(\hat{y})과 실제값(y) 사이의 오차(손실, loss)를 줄이기 위해 각 가중치(w)를 어떻게 조정해야 하는지 계산하는 과정입니다.

핵심 원리는 **연쇄 법칙(Chain Rule)**을 사용하여 손실 함수를 각 가중치로 미분한 값, 즉 기울기(**gradient**)를 구하는 것입니다. 이 기울기는 '오차를 가장 빠르게 줄일 수 있는 방향'을 알려줍니다.

역전파의 핵심

역전파는 출력층에서부터 입력층 방향으로, 순전파 과정에서 계산했던 중간값들을 재활용하여 효율적으로 기울기를 계산합니다. 계산된 기울기는 경사 하강법(Gradient Descent)을 통해 가중치를 업데이트하는 데 사용됩니다.

4.3 가중치와 하이퍼파라미터

[title=파라미터 vs. 하이퍼파라미터]

파라미터 (Parameter) 모델이 학습 과정에서 데이터로부터 스스로 학습하는 변수입니다. **가중치(w)**와 **편향(b)**이 여기에 해당합니다.

하이퍼파라미터 (Hyperparameter) 모델이 학습을 시작하기 전에 사용자가 직접 설정해야 하는 값입니다. **학습률(learning rate)**, 은닉층의 수, 뉴런의 수, 옵티마이저 종류 등이 해당됩니다. 이 값들은 모델의 학습 방식과 최종 성능에 큰 영향을 미칩니다.

5 신경망 학습의 핵심: 경사 하강법

경사 하강법은 손실 함수의 기울기를 이용해 점진적으로 손실이 최소가 되는 지점의 파라미터 값을 찾아가는 최적화 알고리즘입니다. 가중치 업데이트 규칙은 다음과 같습니다.

$$w_{\text{new}} = w_{\text{old}} - \alpha \times \nabla J(w)$$

여기서 α 는 **학습률(learning rate)**이며, 한 번의 업데이트에서 얼마나 큰 폭으로 이동할지를 결정하는 중요한 하이퍼파라미터입니다.

5.1 학습률(α)의 중요성

학습률을 어떻게 설정하느냐에 따라 모델의 학습 속도와 안정성이 크게 달라집니다.

학습률이 너무 작은 경우

현상: 가중치 업데이트 폭이 매우 작아 손실이 거의 줄어들지 않고 학습이 정체됩니다.

결과: 최적점에 도달하는 데 시간이 매우 오래 걸리거나, 학습이 조기에 멈춘 것처럼 보일 수 있습니다. 손실 곡선이 거의 수평선을 그립니다.

학습률이 적절한 경우

현상: 손실이 꾸준히 감소하며 안정적으로 최적점을 찾아갑니다.

결과: 가장 효율적으로 모델을 학습시킬 수 있습니다. 손실 곡선이 부드러운 하강 곡선을 그립니다.

학습률이 너무 큰 경우

현상: 업데이트 폭이 너무 커서 최적점을 지나쳐 버리는 **오버슈팅(overshooting)**이 발생합니다.

결과: 손실 값이 줄어들지 않고 진동하거나 오히려 발산(divergence)하여 학습이 실패할 수 있습니다. 손실 곡선이 위아래로 크게 요동치거나 급격히 증가합니다.

데이터 스케일링의 중요성

경사 하강법이 잘 동작하려면 입력 피처들의 스케일을 비슷하게 맞춰주는 것이 매우 중요합니다. 예를 들어 어떤 피처는 0 1 사이의 값을 갖고, 다른 피처는 100만 단위의 값을 갖는다면 학습이 불안정해집니다. **표준화(Standardization)**나 **정규화(Normalization)**를 통해 모든 입력 데이터의 범위를 비슷하게 만들어주면, 기본 학습률 값으로도 안정적인 학습이 가능해집니다.

[title=옵티마이저 (Optimizer)] Adam, RMSprop과 같은 고급 옵티마이저는 학습 과정에서 학습률을 자동으로 조절해주는 기능을 포함하고 있습니다. 이를 통해 사용자가 학습률을 직접 세밀하게 튜닝하는 수고를 덜어주고, 더 빠르고 안정적인 수렴을 돕습니다.

Part III

순차 데이터 처리를 위한 순환 신경망(RNN)

6 순환 신경망(RNN)의 등장 배경

문장, 주가 데이터, 음성 신호와 같이 시간적 순서가 중요한 데이터를 순차 데이터(Sequential Data)라고 합니다. 기존의 완전 연결 신경망(Fully Connected Neural Network)은 각 입력이 독립적이라고 가정하기 때문에, 데이터의 순서 정보를 효과적으로 처리하기 어렵습니다.

순환 신경망(Recurrent Neural Network, RNN)은 이러한 한계를 극복하기 위해 설계되었습니다. RNN은 내부에 '기억' 혹은 '상태(state)'를 유지하는 순환 구조를 가지고 있어, 이전 시점(time step)의 정보를 현재 시점의 계산에 반영할 수 있습니다.

RNN의 핵심 아이디어

RNN은 이전 단계의 출력을 현재 단계의 입력으로 다시 사용하는 **되먹임 루프(feedback loop)** 구조를 가집니다. 이를 통해 순차적인 정보의 흐름을 모델링하고, 시간적 의존성을 학습할 수 있습니다.

7 기본 RNN (Vanilla RNN) 구조

7.1 RNN의 작동 원리

RNN은 순차 데이터의 각 요소를 시간 단계별로 처리합니다.

1. **t=1 시점**: 첫 번째 입력(x_1)과 초기 은닉 상태(h_0 , 보통 0으로 설정)를 받아 첫 번째 은닉 상태(h_1)를 계산합니다.
2. **t=2 시점**: 두 번째 입력(x_2)과 이전 은닉 상태(h_1)를 받아 두 번째 은닉 상태(h_2)를 계산합니다.
3. 이 과정을 데이터의 마지막 요소까지 반복합니다.

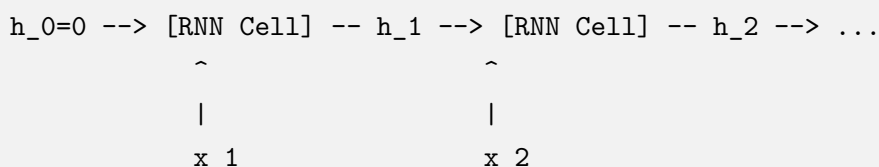
각 시점 t 에서의 은닉 상태 h_t 는 다음과 같이 계산됩니다.

$$h_t = f(W_h h_{t-1} + W_x x_t + b)$$

여기서 f 는 활성화 함수(주로 tanh)이며, W_h , W_x , b 는 모든 시간 단계에서 동일하게 사용되는 **공유 파라미터(shared parameters)**입니다. 이 파라미터 공유 덕분에 RNN은 입력 시퀀스의 길이에 관계없이 모델을 학습시킬 수 있습니다.

□ 예제: title

RNN의 순환 구조는 시간의 흐름에 따라 네트워크를 길게 펼쳐놓은 형태로 시각화할 수 있습니다. 이는 마치 동일한 구조의 네트워크 층이 시간 단계별로 연결된 것처럼 보입니다.



각 [RNN Cell]은 동일한 가중치(W_h, W_x)를 공유합니다.

7.2 RNN의 파라미터 수 계산

하나의 RNN 층(layer)에 있는 파라미터의 수는 입력 벡터의 차원, 은닉 상태 벡터의 차원, 그리고 뉴런의 수에 따라 결정됩니다.

계산 공식:

$$\begin{aligned} \text{파라미터 수} = & (\text{입력 차원} \times \text{뉴런 수}) && - \text{입력 가중치 } W_x \\ & + (\text{이전 은닉 상태 차원} \times \text{뉴런 수}) && - \text{순환 가중치 } W_h \\ & + \text{뉴런 수} && - \text{편향 } b \end{aligned}$$

□ 예제: title

입력 피처가 2개(x_t 가 2차원 벡터)이고, RNN 층에 3개의 뉴런이 있다고 가정해 봅시다. 이 경우 은닉 상태 h_t 는 3차원 벡터가 됩니다.

- 입력 가중치: $2 \times 3 = 6$ 개
- 순환 가중치: $3 \times 3 = 9$ 개
- 편향: 3 개
- 총 파라미터 수: $6 + 9 + 3 = 18$ 개

8 RNN의 한계: 기울기 문제

RNN은 이론적으로는 긴 시퀀스를 처리할 수 있지만, 실제 학습 과정에서는 심각한 문제에 직면합니다. 바로 역전파 과정에서 발생하는 기울기 소실(**Vanishing Gradient**)과 기울기 폭주(**Exploding Gradient**) 문제입니다.

8.1 기울기 소실 (Vanishing Gradient)

원인 역전파 시 기울기는 연쇄 법칙에 따라 여러 번의 곱셈 연산을 거칩니다. RNN에서는 활성화 함수(예: \tanh)의 미분값이 1보다 작은 경우가 많기 때문에, 이 값들이 시간 단계를 거슬러 올라가며 반복적으로 곱해지면 기울기가 기하급수적으로 작아져 거의 0에 가까워집니다.

결과 시퀀스의 앞부분에 있는 정보로부터 온 기울기가 거의 사라져, 해당 정보가 모델 파라미터 업데이트에 거의 영향을 미치지 못하게 됩니다. 이로 인해 RNN은 문장의 시작 부분 단어나 오래전의 추가 데이터와 같은 장기 의존성(long-term dependency)을 학습하기 매우 어려워집니다.

8.2 기울기 폭주 (Exploding Gradient)

원인 기울기 소실과 반대로, 미분값이 1보다 큰 값들이 반복적으로 곱해지면 기울기가 기하급수적으로 커져 무한대에 가까워질 수 있습니다.

결과 파라미터 업데이트가 비정상적으로 크게 일어나 모델 학습이 불안정해지고, 결국 발산하게 됩니다. 이 문제는 기울기 클리핑(**Gradient Clipping**)이라는 기법을 통해, 기울기 값이 특정 임계치를 넘으면 강제로 잘라내어 어느 정도 해결할 수 있습니다.

기본 RNN의 근본적 한계

기울기 소실 문제 때문에, 기본 RNN(Vanilla RNN)은 실제 문제에서 긴 시퀀스의 의존 관계를 효과적으로 학습하지 못합니다. 이러한 한계를 극복하기 위해 LSTM과 GRU 같은 개선된 구조가 제안되었습니다.

Part IV

장기 의존성 문제 해결을 위한 고급 RNN 모델

9 LSTM (Long Short-Term Memory)

LSTM은 기본 RNN의 장기 의존성 학습 문제를 해결하기 위해 설계된 정교한 구조입니다. 핵심은 셀 상태(Cell State)와 3개의 게이트(Gate)를 도입하여 정보의 흐름을 효과적으로 제어하는 데 있습니다.

LSTM의 핵심 비유

LSTM의 셀 상태(c_t)를 '정보 고속도로'라고 생각할 수 있습니다. 이 고속도로를 따라 정보가 거의 변하지 않고 쭉 전달될 수 있어, 오래전 정보도 손실 없이 보존됩니다. 게이트들은 이 고속도로에 정보를 올리거나(입력 게이트), 내리거나(삭제 게이트), 또는 현재 정보를 외부에 보여줄지(출력 게이트)를 결정하는 '톨게이트' 역할을 합니다.

9.1 LSTM의 주요 구성 요소

셀 상태 (Cell State, c_t) LSTM의 핵심으로, 장기 기억을 담당합니다. 시퀀스를 따라 정보가 큰 변화 없이 전달될 수 있는 통로 역할을 합니다.

삭제 게이트 (Forget Gate) 이전 셀 상태(c_{t-1})에서 어떤 정보를 잊어버릴지(버릴지) 결정합니다. 시그모이드 함수를 통해 0(모두 잊음)에서 1(모두 기억) 사이의 값을 출력하여, 이전 정보에 곱해집니다.

입력 게이트 (Input Gate) 새로운 입력 정보(x_t) 중 어떤 것을 셀 상태에 저장할지 결정합니다. 시그모이드 함수로 저장할 정보의 비율을 정하고, tanh 함수로 새로운 후보 정보를 생성한 뒤 두 값을 곱해 셀 상태에 더합니다.

출력 게이트 (Output Gate) 현재 셀 상태를 바탕으로 어떤 정보를 외부로 출력하고, 다음 시점의 은닉 상태(h_t)로 넘겨줄지를 결정합니다.

이러한 게이트 구조 덕분에 LSTM은 기울기 소실 문제에 훨씬 강인하며, 수백 개의 시간 단계를 넘어서는 장기 의존성도 성공적으로 학습할 수 있습니다.

10 GRU (Gated Recurrent Unit)

GRU는 LSTM의 복잡한 구조를 간소화하면서도 유사한 성능을 내는 모델입니다. LSTM의 셀 상태와 은닉 상태를 하나의 은닉 상태(h_t)로 통합하고, 게이트 수도 2개로 줄였습니다.

리셋 게이트 (Reset Gate) 이전 은닉 상태의 어느 부분을 무시할지를 결정합니다. 이는 새로운 입력을 기반으로 한 새로운 기억을 만드는 데 영향을 줍니다.

업데이트 게이트 (Update Gate) LSTM의 삭제 게이트와 입력 게이트 역할을 동시에 수행합니다. 이전 상태의 정보를 얼마나 유지하고, 새로운 정보를 얼마나 반영할지 결정합니다.

[title=LSTM vs. GRU] 구조: LSTM은 3개의 게이트와 별도의 셀 상태를 가지지만, GRU는 2개의 게이트와 통합된 은닉 상태를 가집니다.

파라미터: GRU가 LSTM보다 파라미터 수가 적어 계산 효율이 높고, 데이터가 적을 때 과적합의 위험이 적을 수 있습니다.

성능: 대부분의 문제에서 두 모델의 성능은 비슷하지만, 문제의 특성이나 데이터의 양에 따라 어느 한쪽이 더 나을 수 있습니다. 일반적으로는 LSTM이 더 복잡한 패턴을 학습할 수 있는 잠재력이 있습니다.

11 양방향 RNN (Bidirectional RNN)

문장 번역이나 감성 분석과 같은 문제에서는 특정 단어의 의미가 앞뒤 문맥에 모두 의존하는 경우가 많습니다. 예를 들어, "나는 그 영화가 정말 재미____"라는 문장에서 마지막 단어를 예측하려면 앞의 내용뿐만 아니라, 문장 끝에 "없었다"가 오는지 "있었다"가 오는지를 알아야 합니다.

기본적인 RNN은 과거 정보만을 이용하므로 이러한 양방향 문맥을 파악할 수 없습니다. **양방향 RNN(Bidirectional RNN, Bi-RNN)**은 이 문제를 해결하기 위해 고안되었습니다.

11.1 Bi-RNN의 구조와 원리

Bi-RNN은 내부적으로 두 개의 독립적인 RNN 층을 가집니다.

1. **정방향 RNN (Forward RNN)**: 입력 시퀀스를 원래 순서대로 (예: x_1, x_2, \dots, x_T) 처리합니다.
2. **역방향 RNN (Backward RNN)**: 입력 시퀀스를 역순으로 (예: x_T, \dots, x_2, x_1) 처리합니다.

각 시간 단계 t 에서, Bi-RNN의 최종 출력은 정방향 RNN의 은닉 상태(\vec{h}_t)와 역방향 RNN의 은닉 상태(\overleftarrow{h}_t)를 **연결(concatenate)**하여 만들어집니다.

$$h_t = [\vec{h}_t; \overleftarrow{h}_t]$$

이를 통해 모델은 각 시점에서 과거와 미래의 정보를 모두 활용하여 더 풍부한 문맥적 표현을 학습할 수 있습니다.

□ 예제: title

Keras에서는 Bidirectional 래퍼(wrapper)를 사용하여 간단하게 양방향 모델을 구현할 수 있습니다.

```
1 from keras.models import Sequential
2 from keras.layers import Embedding, LSTM, Dense, Bidirectional
3
4 model = Sequential()
5 model.add(Embedding(input_dim=200, output_dim=32))
6 model.add(Bidirectional(LSTM(32))) # LSTM 층으로 Bidirectional 감싸기
7 model.add(Dense(1, activation='sigmoid'))
8
9 model.summary()
```

Listing 1: Keras를 이용한 양방향 LSTM 모델 구축

위 코드에서 양방향 LSTM 층의 출력 차원은 (None, 64)가 됩니다. 이는 정방향 LSTM(32차원)과 역방향 LSTM(32차원)의 출력이 연결되었기 때문입니다.

Part V

FAQ 및 요약

12 자주 묻는 질문 (FAQ)

Q. 에포크(Epoch), 배치(Batch), 이터레이션(Iteration)의 차이는 무엇인가요?

A. 세 용어는 모델 학습 단위를 나타냅니다.

- **에포크 (Epoch):** 전체 학습 데이터셋을 한 번 모두 사용했을 때 1 에포크가 됩니다.
- **배치 (Batch):** 전체 데이터를 한 번에 처리하기엔 너무 크므로, 작은 묶음으로 나눕니다. 이 묶음 하나를 배치라고 합니다.
- **이터레이션 (Iteration):** 하나의 배치를 처리하여 가중치를 한 번 업데이트하는 것을 1 이터레이션이라고 합니다. 즉, (총 데이터 수 / 배치 크기) 만큼의 이터레이션이 1 에포크가 됩니다.

Q. 학습 정확도(Training Accuracy)는 계속 오르는데, 검증 정확도(Validation Accuracy)는 정체되거나 떨어집니다. 왜 그런가요?

A. 이는 모델이 학습 데이터에만 너무 과하게 맞춰져 새로운 데이터에 대한 일반화 성능이 떨어지는 **과적합(Overfitting)**의 전형적인 신호입니다. 모델이 데이터의 실제 패턴이 아닌 노이즈까지 암기하기 시작했다는 의미입니다. 이 경우, 검증 정확도가 가장 높았던 시점에서 학습을 조기 종료(Early Stopping)하거나, 드롭아웃(Dropout)과 같은 규제(Regularization) 기법을 적용해야 합니다.

Q. 데이터를 스케일링(scaling)하는 것이 왜 중요한가요?

A. 신경망의 학습 알고리즘인 경사 하강법은 각 피처(feature)의 스케일에 민감합니다. 만약 피처들의 값 범위가 크게 다르면(예: 나이는 0-100, 소득은 수천만 단위), 손실 함수의 표면이 한쪽으로 길게 찌그러진 타원형이 됩니다. 이런 경우 최적점을 찾아가는 경로가 매우 비효율적인 지그재그 형태가 되어 학습이 느려지거나 불안정해집니다. 모든 피처의 스케일을 비슷하게 맞춰주면(예: 0 1 사이로 정규화) 손실 함수 표면이 원형에 가까워져, 훨씬 빠르고 안정적으로 최적점을 찾을 수 있습니다.

13 한 페이지 요약

TF-IDF

문서 내 단어의 중요도를 평가하는 지표. **단어 빈도(TF)**와 **역문서 빈도(IDF)**의 곱으로 계산. 문서의 핵심 키워드를 추출하고 텍스트를 벡터로 변환하는 데 사용.

순전파 & 역전파

순전파: 입력에서 출력으로 값을 계산하는 과정. **역전파:** 출력의 오차를 바탕으로, 가중치를 업데이트하기 위해 기울기를 계산하는 과정.

학습률 (Learning Rate)

경사 하강법에서 가중치를 업데이트하는 보폭(step size). 너무 작으면 학습이 느리고, 너무 크면 학습이 발산할 수 있음. Adam과 같은 옵티마이저는 이를 자동으로 조절.

순환 신경망 (RNN)

순서가 있는 데이터를 처리하기 위한 신경망. 이전 시점의 은닉 상태(hidden state)를 현재 계산에 활용하여 시간적 의존성을 학습. 가중치 공유가 특징.

기울기 소실 & 폭주

RNN의 고질적인 문제. 긴 시퀀스에서 역전파 시 기울기가 0 또는 무한대로 수렴하여 장기 의존성 학습을 방해함.

LSTM & GRU

기울기 소실 문제를 해결하기 위한 RNN의 변형. 게이트(Gate) 메커니즘을 도입하여 정보의 흐름을 제어하고, 장기 기억을 효과적으로 보존함.

양방향 RNN (Bi-RNN)

시퀀스를 정방향과 역방향으로 모두 처리하여 각 시점에서 과거와 미래의 문맥을 모두 활용하는 모델. 언어 처리와 같이 양방향 문맥이 중요한 작업에서 성능이 높음.