Text Classification

Thomas Bayes - Image from Wikipedia

Text Classification

Text Classification(텍스트 분류)은 텍스트, 문장 또는 문서(문장들)를 입력으로 받아 사전에 정의된 클래스(class)들 중에서 어떤 클래스에 속하는지 분류 하는 과정을 의미합니다. 따라서 Text Classification은 어쩌면 이 책에서 (그 난이도에 비해서) 독자들에게 가장 쓸모가 있는 챕터가 될 수도 있습니다. Text Classificaion의 응용 분야가 다양하기 때문 입니다.

문제	클래스 예
감성분석(Sentiment Analysis)	긍정(positive), 중립(neutral), 부정(negative)
스팸 메일 탐지(Spam E-mail Detection)	정상(normal), 스팸(spam)
사용자 의도 분류(User Intent Classificaion)	명령, 질문, 잡담 등
주제 분류	각 주제
카테고리 분류	각 카테고리

등 무언가 분류해야 하는 문제가 있다면 대부분 text classificaion에 속한다고 볼 수 있습니다. 딥러닝 이전에는 Naive Bayes, SVM 등 다양한 방법이 존재하였습니다. 이번 챕터에서는 딥러닝 이전의 가장 간단한 방식인 naive bayes 방식과 딥러닝 방식들을 소개 하도록 하겠습니다. # Naive Bayes

Naive Bayes는 매우 간단하지만 정말 강력한 방법 입니다. 의외로 기대 이상의 성능을 보여줄 때가 많습니다. 물론 단어를 여전히 discrete한 심볼로 다루기 때문에, 여전히 아쉬운 부분이 많습니다. 이번 섹션에서는 Naive Bayes를 통해서 텍스트 분류를 하는 방법을 살펴 보겠습니다.

Maximum A Posterior

Naive Bayes를 소개하기에 앞서 Bayes Theorem(베이즈 정리)을 짚고 넘어가지 않을 수 없습니다. Thomas Bayes(토마스 베이즈)가 정립한 이 정리에 따르면 조건부 확률은 아래와 같이 표현 될 수 있으며, 각 부분은 명칭을 갖고 있습니다. 이 이름들에 대해서는 앞으로 매우 친숙해져야 합니다.

$$\underbrace{P(Y|X)}_{posterior} = \frac{\overbrace{P(X|Y)}^{likelihood}\overbrace{P(Y)}^{prior}}{\underbrace{P(X)}_{evidence}}$$

수식	영어 명칭	한글 명칭
P(Y|X)	Posterior	사후 확률
P(X|Y)	Likelihood	가능도(우도)
P(Y)	Prior	사전 확률
P(X)	Evidence	증거

우리가 풀고자하는 대부분의 문제들은 $P(X)$는 구하기 힘들기 때문에, 보통은 아래와 같이 접근 하기도 합니다.

$$P(Y|X) \propto P(X|Y)P(Y)$$

위의 성질을 이용하여 주어진 데이터 X를 만족하며 확률을 최대로 하는 클래스 Y를 구할 수 있습니다. 이처럼 posterior 확률을 최대화(maximize)하는 y를 구하는 것을 Maximum A Posterior (MAP)라고 부릅니다. 그 수식은 아래와 같습니다.

$$\hat{y}_{MAP} = argmax_{y \in \mathcal{Y}} P(Y = y|X)$$

다시한번 수식을 살펴보면, X(데이터)가 주어졌을 때, 가능한 클래스의 set $mathcalY$ 중에서 posterior를 최대로 하는 클래스 y를 선택하는 것 입니다.

이와 마찬가지로 X(데이터)가 나타날 likelihood 확률을 최대로 하는 클래스 y를 선택하는 것을 Maximum Likelihood Estimation (MLE)라고 합니다.

$$\hat{y}_{MLE} = argmax_{y \in \mathcal{Y}} P(X|Y = y)$$

MLE는 주어진 데이터X와 클래스 레이블(label) Y가 있을 때, parameter θ를 훈련하는 방법으로도 많이 사용 됩니다.

$$\hat{\theta} = argmax_{\theta} P(Y|X, \theta)$$

MLE vs MAP

경우에 따라 MAP는 MLE에 비해서 좀 더 정확할 수 있습니다. prior(사전)확률이 반영되어 있기 때문 입니다. 예를 들어보죠.

만약 범죄현장에서 발자국을 발견하고 사이즈를 측정했더니 범인은 신발사이즈(데이터, X) 155를 신는 사람인 것으로 의심 됩니다. 이때, 범인의 성별(클래스, Y)을 예측해 보도록 하죠.

성별 클래스의 set은 $Y = \{male, female\}$ 입니다. 신발사이즈 X는 5단위의 정수로 이루어져 있습니다. $X = \{\cdots, 145, 150, 155, 160, \cdots\}$

신발사이즈 155는 남자 신발사이즈 치곤 매우 작은 편 입니다. 따라서 우리는 보통 범인을 여자라고 특정할 것 같습니다. 다시 말하면, 남자일 때 신발사이즈 155일 확률 $P(X = 155|Y = male)$은 여자일 때 신발사이즈 155일 확률 $P(X = 155|Y = female)$일 확률 보다 낮습니다.

보통의 경우 남자와 여자의 비율은 0.5로 같기 때문에, 이는 큰 상관이 없는 예측입니다. 하지만 범죄현장이 만약 군부대였다면 어떻게 될까요? 남녀 성비는 $P(Y = male) >> P(Y = female)$로 매우 불균형 할 것입니다.

이때, 이미 갖고 있는 likelihood에 prior를 곱해주면 posterior를 최대화 하는 클래스를 더 정확하게 예측 할 수 있습니다.

$$P(Y = male|X = 155) \propto P(X = 155|Y = male)P(Y = male)$$

Naive Bayes

Naive Bayes는 MAP를 기반으로 동작합니다. 대부분의 경우 posterior를 바로 구하기 어렵기 때문에, likelihood와 prior의 곱을 통해 클래스 Y를 예측 합니다.

이때, X가 다양한 feature(특징)들로 이루어진 데이터라면, 훈련 데이터에서 매우 희소(rare)할 것이므로 likelihood $P(X = w_1, w_2, \cdots, w_n|Y = c)$를 구하기 어려울 것 입니다. 이때 Naive Bayes가 강력한 힘을 발휘 합니다. 각 feature들이 상호 독립적이라고 가정하는 것 입니다. 그럼 joint probability를 각 확률의 곱으로 근사(approximate)할 수 있습니다. 이 과정을 수식으로 표현하면 아래와 같습니다.

$$\begin{aligned} P(Y = c|X = w_1, w_2, \cdots, w_n) &\propto P(X = w_1, w_2, \cdots, w_n|Y = c)P(Y = c) \\ &\approx P(w_1|c)P(w_2|c)\cdots P(w_n|c)P(c) \\ &= \prod_{i=1}^{n} P(w_i|c)P(c) \end{aligned}$$

따라서, 우리가 구하고자 하는 MAP를 활용한 클래스는 아래와 같이 posterior를 최대화하는 클래스가 되고, 이는 Naive Bayes의 가정에 따라 각 feature들의 확률의 곱에 prior확률을 곱한 값을 최대화 하는 클래스와 같을 것 입니다.

$$\hat{c}_{MAP} = argmax_{c \in \mathcal{C}} P(Y = c | X = w_1, w_2, \cdots, w_n)$$
$$= argmax_{c \in \mathcal{C}} \prod_{i=1}^{n} P(w_i|c) P(c)$$

이때 사용되는 prior 확률은 아래와 같이 실제 데이터에서 나타난 횟수를 세어 구할 수 있습니다.

$$\tilde{P}(Y = c) = \frac{Count(c)}{\sum_{i=1}^{|\mathcal{C}|} Count(c_i)}$$

또한, 각 feature 별 likelihood 확률도 데이터에서 바로 구할 수 있습니다. 만약 모든 feature들의 조합이 데이터에서 나타난 횟수를 통해 확률을 구하려 하였다면 sparseness(희소성) 문제 때문에 구할 수 없었을 것 입니다. 하지만 Naive Bayes의 가정(각 feature들은 독립적)을 통해서 쉽게 데이터에서 출현 빈도를 활용할 수 있게 되었습니다.

$$\tilde{P}(w|c) = \frac{Count(w, c)}{\sum_{j=1}^{|V|} Count(w_j, c)}$$

이처럼 간단한 가정을 통하여 데이터의 sparsity를 해소하여, 간단하지만 강력한 방법으로 우리는 posterior를 최대화하는 클래스를 예측 할 수 있게 되었습니다.

Example: Sentiment Analysis

그럼 실제 예제로 접근해 보죠. 감성분석은 가장 많이 활용되는 텍스트 분류 기법입니다. 사용자의 댓글이나 리뷰 등을 긍정 또는 부정으로 분류하여 마케팅이나 서비스 향상에 활용하고자 하는 방법 입니다. 물론 실제로 딥러닝 이전에는 Naive Bayes를 통해 접근하기보단, 각 클래스 별 어휘 사전(vocabulary)을 만들어 해당 어휘의 등장 여부에 따라 판단하는 방법을 주로 사용하곤 하였습니다.

$$\mathcal{C} = \{pos, neg\}$$
$$\mathcal{D} = \{d_1, d_2, \cdots\}$$

위와 같이 긍정(pos)과 부정(neg)으로 클래스가 구성($\mathcal{C}$되어 있고, 문서 d로 구성된 데이터 $\mathcal{D}$가 있습니다.

이때, 우리에게 "I am happy to see this movie!"라는 문장이 주어졌을 때, 이 문장이 긍정인지 부정인지 판단해 보겠습니다.

$$P(pos|I, am, happy, to, see, this, movie, !) = \frac{P(I, am, happy, to, see, this, movie, !|pos)P(pos)}{P(I, am, happy, to, see, this, movie, !)}$$
$$\approx \frac{P(I|pos)P(am|pos)P(happy|pos)\cdots P(!|pos)P(pos)}{P(I, am, happy, to, see, this, movie, !)}$$

Naive Bayes의 수식을 활용하여 단어의 조합에 대한 확률을 각각 분해할 수 있습니다. 그리고 그 확률들은 아래와 같이 데이터 $\mathcal{D}$에서의 출현 빈도를 통해 구할 수 있습니다.

$$P(happy|pos) \approx \frac{Count(happy, pos)}{\sum_{j=1}^{|V|} Count(w_j, pos)}$$
$$P(pos) \approx \frac{Count(pos)}{|\mathcal{D}|}$$

마찬가지로 부정 감성에 대해 같은 작업을 반복 할 수 있습니다.

$$P(neg|I, am, happy, to, see, this, movie, !) = \frac{P(I, am, happy, to, see, this, movie, !|neg)P(neg)}{P(I, am, happy, to, see, this, movie, !)}$$
$$\approx \frac{P(I|neg)P(am|neg)P(happy|neg)\cdots P(!|neg)P(neg)}{P(I, am, happy, to, see, this, movie, !)}$$

$$P(happy|neg) \approx \frac{Count(happy, neg)}{\sum_{j=1}^{|V|} Count(w_j, neg)}$$
$$P(neg) \approx \frac{Count(neg)}{|\mathcal{D}|}$$

Add-one Smoothing

여기에 문제가 하나 있습니다. 만약 훈련 데이터에서 $Count(happy, neg)$가 0이었다면 $P(happy|neg) = 0$이 되겠지만, 그저 훈련 데이터에 존재하지 않는 경우라고 해서 실제 출현 확률을 0으로 여기는 것은 매우 위험한 일 입니다.

$$P(happy|neg) \approx \frac{Count(happy, neg)}{\sum_{j=1}^{|V|} Count(w_j, neg)} = 0,$$

$$\text{where } Count(happy, neg) = 0.$$

따라서 우리는 이런 경우를 위하여 각 출현횟수에 1을 더해주어 간단하게 문제를 완화할 수 있습니다. 물론 완벽한 해결법은 아니지만, Naive Bayes의 가정과 마찬가지로 간단하고 강력합니다.

$$\tilde{P}(w|c) = \frac{Count(w, c) + 1}{\left(\sum_{j=1}^{|V|} Count(w_j, c)\right) + |V|}$$

Conclusion

위와 같이 Naive Bayes를 통해서 단순히 출현빈도를 세는 것처럼 쉽고 간단하지만 강력하게 감성분석을 구현 할 수 있습니다. 하지만 문장 "I am not happy to see this movie!"라는 문장이 주어지면 어떻게 될까요? "not"이 추가 되었을 뿐이지만 문장의 뜻은 반대가 되었습니다.

$$P(pos|I, am, not, happy, to, see, this, movie, !)$$
$$P(neg|I, am, not, happy, to, see, this, movie, !)$$

"not"은 "happy"를 수식하기 때문에 두 단어를 독립적으로 보는 것은 옳지 않을 수 있습니다.

$$P(not, happy) \neq P(not)P(happy)$$

사실 문장은 단어들이 순서대로 나타나서 의미를 이루기 때문에, 각 단어의 출현 여부도 중요하지만, 각 단어 사이의 순서로 인해 생기는 관계도 무시할 수 없습니다. 하지만 Naive Bayes의 가정은 언어의 이런 특징을 단순화하여 접근하기 때문에 한계가 있습니다.

하지만, 레이블(labeled) 데이터가 매우 적은 경우에는 딥러닝보다 이런 간단한 방법을 사용하는 것이 훨씬 더 나은 대안이 될 수도 있습니다. 이처럼 Naive Bayes는 매우

간단하고 강력하지만, Naive Bayes를 강력하게 만들어준 가정이 가져오는 단점 또한 명확합니다. # CNN을 통한 접근 방법

[Kim at el.2014] article from WildML

CNN 개념 설명

How to use CNN to NLP

구조 설계

설명

코드

RNN을 통한 접근 방법

구조 설계

설명

코드

Unsupervised Text Classification

[Radford at el.2017]

소개

구조 설계

설명

코드