

Lecture 08

Dihui Lai

dlai@wustl.edu

March 17, 2020

Multi-class Classification

Multinomial Distribution

Multinomial distribution: there could be c outcome of an experiment, each of probability $p_1, p_2, p_3, \dots, p_c$ and $\sum_{k=1}^c p_k = 1$. If one perform M experiments, the probability of getting $m_1, m_2, m_3, \dots, m_c$ of each outcome can be described as

$$f(m_1, m_2, m_3, \dots, m_c) = \frac{M!}{m_1! m_2! \dots m_c!} p_1^{m_1} p_2^{m_2} \dots p_c^{m_c}$$

Multi-class Classification

If a target variable contains more than 2 types of outcome, logistic regression can not handle it easily. We need to build a **multi-class classification** model

Assume there can be C possible outcome in your target variable, we can use one-hot encoding and denote the outcome of i^{th} data point $\vec{y}^i = [y_1^i, y_2^i, \dots, y_C^i]$, only one element out of C is 1.

Multi-class Classification: Likelihood Function

For each data point i , the likelihood function can be constructed as

$L^i = \prod_{k=1}^C p_k^i y_k^i$ i.e. multinomial distribution of ($M=1$). For an outcome of class c , we have $y_c = 1$ and all other elements of y are 0. $y_1 = 0, y_2 = 0, \dots, y_C = 0$. The log-likelihood of the data point is $\ell^i = \sum_{k=1}^C y_k^i \log(p_k^i)$

The log-likelihood function (a.k.a log-loss) is

$$\ell = \log(L) = \sum_{i=1}^n \sum_{k=1}^C y_k^i \log(p_k^i)$$

Multi-Class classification: Softmax

To estimate p_k^i , we use softmax function of $\vec{x} \cdot \vec{\beta}_k$

$$p_k^i = \frac{e^{\vec{x}^i \cdot \vec{\beta}_k}}{\sum_{k=1}^C e^{\vec{x}^i \cdot \vec{\beta}_k}}$$

Multi-Class classification: Optimization

To find the $\vec{\beta}_k$, we need to solve

$$\frac{\partial \ell}{\partial \beta_{kj}} = 0$$

Python method: `"statsmodels.api.MNLogit"`

Naive Bayes Classifier

Bayes' Theorem

Bayes' Theorem: given two random variables X and Y , the conditional probability of X given Y is expressed as:

$$P(Y | X) = \frac{P(X | Y)P(Y)}{P(X)}$$

Useful terminologies to interpret the equations: $P(Y)$ is called prior, which is the belief in Y without any other knowledge. $P(Y | X)$ is the posterior taking into consideration of X . $P(X | Y)$ is the likelihood.

In a discrete case, the probability distribution of X can be calculated as $P(X) = \sum_i P(X | Y_i)P(Y_i)$

Baye's Theorem Example

Rain in California

You are planning a trip to california tomorrow. Unfortunately, the weatherman has predicted rain for tomorrow. You know in southern california, it only rains 5 days each year and there is a chance the weather man makes false predictions. You searched on line and find that when it rains, the weatherman correctly forecasts rain of 90% of the time. When it doesn't rain, he incorrectly forecasts rain 10% of the time. What is the probability that it will rain tomorrow.

Baye's Theorem Example

Solution: Denote the event that the weatherman forecast a raining day as F . The probability of rain given weatherman's forecast is

$$P(1 | F) = \frac{P(1)P(F | 1)}{P(F)}$$

Given that we have $P(F | 1) = 0.9$, $P(F | 0) = 0.1$, $P(1) = \frac{5}{365}$ and $P(0) = 1 - P(1) = \frac{360}{365}$. $P(F) = P(F|1)P(1) + P(F|0)P(0)$, Therefore,

$$P(1 | F) = \frac{P(1)P(F | 1)}{P(F)} = \frac{\frac{5}{365} \cdot 0.9}{0.1109} = 0.111$$

Joint Probability Distribution and Chain Rule

- The joint probability distribution of two events can be described as

$$P(A, B) = P(A | B)P(B) = P(B | A)P(A)$$

If A and B are two independent events, then we have

$$P(B) = P(B | A) \text{ and } P(A) = P(A | B)$$

- Chain Rule:** Considering n random events $X_1, X_2, X_3 \dots X_n$, their joint probability distribution can be described as

$$\begin{aligned} P(X_n, \dots, X_1) \\ &= P(X_n | X_{n-1}, \dots, X_1) P(X_{n-1}, \dots, X_1) \\ &= P(X_n | X_{n-1}, \dots, X_1) P(X_{n-1} | X_{n-2}, \dots, X_1) P(X_{n-2}, \dots, X_1) \\ &= \dots \end{aligned}$$

Naive Bayes Classifier

The joint probability distribution of predictor \vec{x} and target variable y can be written as

$$\begin{aligned}
 & p(x_1, x_2, \dots, x_n, y) \\
 &= p(x_1 | x_2, x_3, \dots, y) p(x_2, x_3, \dots, y) \\
 &= p(x_1 | x_2, x_3, \dots, y) p(x_2 | x_3, x_4, \dots, y) p(x_3, x_4, \dots, y) \\
 &= p(x_1 | x_2, x_3, \dots, y) p(x_2 | x_3, x_4, \dots, y) \dots p(x_{n-1} | x_n, y) p(x_n | y) p(y)
 \end{aligned}$$

Assuming features are independent of each other but only dependent on the target variable, then we have

$$p(x_1, x_2, \dots, x_n, y) = p(y) \prod_{i=1}^N p(x_i | y)$$

Naive Bayes Classifier

Using Bayes' theorem, we can get the conditional probability distribution of the target variable as

$$p(Y|X) = \frac{p(X, Y)}{p(X)}$$

Therefore, we have

$$p(Y|X) = \frac{p(y) \prod_{i=1}^N p(x_i|y)}{\sum_y p(y) \prod_{i=1}^N p(x_i|y)}$$

The denominator is constant if the features are known. $p(y)$ and $p(x_i | y)$ can be calculated from the data. We need to find the y that maximizes the $p(Y | X)$, i.e.

$$\hat{y} = \underset{y}{\operatorname{argmax}} p(y) \prod_{i=1}^N p(x_i|y)$$

Nature Language Process

Word Semantics and Representations

- Homonymous: a word can have multiple definitions e.g. mouse could mean small rodents or it could mean computer devices.
- Synonyms/antonym (words' relations): couch/sofa, vomit/throw up, filbert/hazelnut; long/short, big/little
- Word sentiments
- Can we represent a word using vectors and quantify those measures?

Word Vector Representations

Term-term matrix or word-word matrix: count the number of times a word occurs in a context window around the target word (e.g. ± 7)
 sugar, a sliced lemon, a tablespoonful of, **apricot** jam, a pinch each of,

	aardvark	...	computer	data	pinch	result	sugar	..
apricot	0	...	0	0	1	0	1	..
pineapple	0	...	0	0	1	0	1	..
digital	0	...	2	1	0	1	0	..
information	0	...	1	6	0	4	0	..

It can be inferred from the word-word matrix that apricot and pineapple are more similar to each other.

Cosine Similarity

The similarity of two words could be measured by dot-products of their vector representation

$$\vec{v} \cdot \vec{w} = \sum_{i=1}^N v_i w_i$$

The dot-product favors vectors of higher frequency to normalize the similarity without considering word frequency, we use cosine similarity measure

$$\text{cosine}(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}| |\vec{w}|} = \frac{\sum_{i=1}^N v_i w_i}{\sqrt{\sum_{i=1}^N v_i^2} \sqrt{\sum_{i=1}^N w_i^2}}$$

N-gram Language Models

- Models that assign probabilities to sequences of words are called language models or LM.
- An n-gram is a sequence of N words e.g. 2-gram (or bigram) "Good Morning", 3-gram "Turn it on"
- N-gram language models estimate the probability of the last word of an n-gram given the previous words

N-gram Language Models

LM: What is the probability of having a sentence that consists a sequence of words: $w_1, w_2, w_3 \dots w_N$, i.e. $P(w_1, w_2, w_3 \dots w_N)$.

Recall the chain rule:

$$\begin{aligned} P(w_1, w_2, w_3 \dots w_N) \\ = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)P(w_4|w_1, w_2, w_3) \dots P(w_N|w_1, w_2, \dots w_{N-1}) \end{aligned}$$

In the case of bigram, we assume $P(w_N|w_1, \dots, w_{N-1}) = P(w_N|w_{N-1})$, since the word is only dependent on the previous word, it is also called Markov assumption. In general case of an n-gram, we assume

$$P(w_N|w_1, w_2, \dots w_{N-1}) = P(w_N|w_{N-1}, w_{N-2}, \dots w_{N-n+1})$$

MLE Estimation for bigram

In the case of bigram, the MLE estimation can be formulated as

$$P(w_N|w_{N-1}) = \frac{C(w_{N-1}w_N)}{\sum_w C(w_{N-1}w)} = \frac{C(w_{N-1}w_N)}{C(w_{N-1})}$$

Here, C is the count of the words' occurrence

Example: MLE Estimation for bigram

Estimate the bigram for the following corpus, here $\langle s \rangle$ and $\langle /s \rangle$ are introduced as the symbols that represents the begining and end of a setence.

$\langle s \rangle$ I am Sam $\langle /s \rangle$

$\langle s \rangle$ Sam I am $\langle /s \rangle$

$\langle s \rangle$ I do not like green eggs and ham $\langle /s \rangle$

We begin buy counting the words occurence and have $C(I) = 3$,
 $C(\text{Sam}) = 2$, $C(\langle /s \rangle) = 3$, $C(\langle s \rangle) = 3 \dots C(\langle s \rangle I) = 2$, $C(\langle s \rangle \text{Sam}) = 1$

So we have $P(I|\langle s \rangle) = \frac{2}{3}$, $P(\text{Sam}|\langle s \rangle) = \frac{1}{3}$, $P(\text{do}|I) = \frac{1}{3}$, $P(\text{am}|I) = \frac{2}{3}$,
 $P(\text{Sam}|\text{am}) = \frac{1}{2}$, $P(\langle /s \rangle|\text{Sam}) = \frac{1}{2}$

The in-sample probability of $P(\langle s \rangle I \text{ am Sam} \langle /s \rangle) =$
 $P(I|\langle s \rangle)P(\text{am}|I)P(\text{Sam}|\text{am})P(\langle /s \rangle|\text{Sam}) = \frac{2}{3} \times \frac{2}{3} \times \frac{1}{2} \times \frac{1}{2}$

Evaluating Language Models

How do we compare two LM?

- A test data/hold out data set can be used to evaluate a LM. Apply the estimated conditional probability to the test data set and compare the resulting probability.
- Perplexity is used instead of the raw probability.

$$\begin{aligned} PP(W) &= P(w_1, w_2, \dots w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1, w_2, \dots w_N)}} \end{aligned}$$

- Maximize probability is equivalent to minimize perplexity

Smoothing

What do we do with words that appear in a test set with an unseen context for example, $P(\text{John}|\text{am}) = 0$ because "John" has never appear in training text. We end up getting $P(w_1, w_2, \dots w_N) = 0$. One possible solution is smoothing

- Laplace smoothing: increase the bigram count by 1, so what was counted 0 now becomes 1

$$P(w_N|w_{N-1}) = \frac{C(w_{N-1}w_N) + 1}{\sum_w C(w_{N-1}w) + 1} = \frac{C(w_{N-1}w_N) + 1}{C(w_{N-1}) + V}$$

The denominator is adjusted by the vocabulary size of V

- Add-k smoothing, increase the count by a fraction of k (0.5, 0.8 ...) and we have

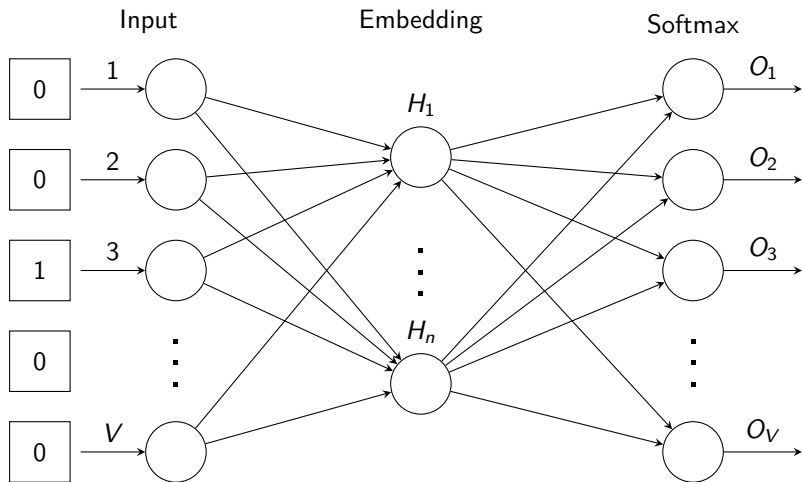
$$P(w_N|w_{N-1}) = \frac{C(w_{N-1}w_N) + k}{\sum_w C(w_{N-1}w) + k} = \frac{C(w_{N-1}w_N) + k}{C(w_{N-1}) + kV}$$

Unknown Words

What do we do if a word in the test data is not in the vocabulary i.e. out of vocabulary (OOV)

- Choose a fixed vocabulary. If a word in the training set is OOV, convert it to $\langle UNK \rangle$. Estimate the probability of $\langle UNK \rangle$ as a regular word.
- replace low frequency word in the training dataset by $\langle UNK \rangle$. Treating $\langle UNK \rangle$ as regular word.

Neural Network Based Language Model: CBOW/Skip-Gram Model



A vocabulary is fed into the neural network using one-hot encoding methods. For a vocabulary of size V , the input vector is of size $1 \times V$.

Neural Network Based Language Model: Architecture

- The input variable is a one-hot encoding vector. If the vocabulary is of size V , an input vector is has V components $\vec{x} = [0, 0, 0 \dots 1, \dots 0]$
- The hidden layer has n neurons. The input weights matrix W is of size $V \times n$
- The output layer weights W' matrix is of size $n \times V$
- CBOW: take $2m$ words (i.e. $w_{c-m}, \dots w_{c-1}, w_{c+1}, w_{c+m}$) around the center word w_c as input w_c is the target.
- Skip-gram: take the center word w_c as the input and the $2m$ words (i.e. $w_{c-m}, \dots w_{c-1}, w_{c+1}, w_{c+m}$) around it as the target.

Reference: <https://arxiv.org/pdf/1301.3781.pdf>

Neural Network Based Language Model: Word Embedding

The word representation/embedding can be calculated as

$$w_i = x_i W$$

x_i is the i^{th} word in the dictionary, w_i is the i^{th} row in the input matrix W