به نام خدا

دانشکده سامانه‌های هوشمند

روش‌های یادگیری ماشین در پردازش زبان طبیعی (۸۳۰۴۳۶۸)

نیم‌سال اول ۱۴۰۴-۱۴۰۳

تمرین شماره ۲

تاریخ تحویل:
۱۴۰۳/۰۸/۱۴

---

۱. (۱۵٪) [نظری – بیز ساده و توزیع احتمالاتی] هدف از این بخش، مطالعه مبحث‌های بیز ساده و توزیع احتمالاتی است.

۱. دسته‌بندی برای تفکیک ایمیل‌های غیر تبلیغاتی از تبلیغاتی طراحی کرده‌ایم، این دسته‌بند در ۹۰ درصد مواقع پاسخ درست می‌دهد. در صورتی که فرض کنیم تنها یک درصد ایمیل‌ها تبلیغاتی هستند، اگر ایمیل تبلیغاتی تشخیص داده شود، با چه احتمالی تبلیغاتی است؟

۲. یک پیشامد موفقیت یا شکست با احتمال موفقیت P مرتبا تکرار می‌شود. پیشامدها در صورت مشاهده‌ی دو موفقیت یا دو شکست پیاپی به پایان می‌رسند. احتمال دو موفقیت پیاپی به شرط آن‌که دو شکست پیاپی اتفاق نیوفتد چیست؟

۳. در یک آزمایش پزشکی سه پارامتر x، y و z اندازه‌گیری می‌شود. اگر شخصی سالم باشد، توزیع احتمالی این سه پارامتر به صورت گوسی با واریانس‌های به ترتیب ۱، ۴ و ۹ و میانگین‌های به ترتیب برابر ۱، ۴ و ۶ است. برای شخص بیمار نیز توزیع احتمالی این سه پارامتر به صورت گوسی با واریانس‌های به ترتیب ۱، ۴ و ۹ و میانگین‌هایی به ترتیب برابر با ۶، ۸ و ۱۰ است. با فرض اینکه قصد داریم از روش بیز ساده (روشی که در آن فرض می‌شود سه پارامتر x، y و z مستقل از یکدیگر هستند) برای تفسیر نتایج این آزمایش استفاده کنیم، نشان دهید این روش معادل این است که حاصل عبارت $\alpha x + \beta y + \gamma z + \lambda$ را بدست آورده و اگر این حاصل مثبت بود، فرد بیمار و در غیر اینصورت فرد سالم در نظر گرفته می‌شود. با فرض اینکه احتمال بیمار بودن فردی که به طور تصادفی از جامعه انتخاب می‌گردد برابر $\frac{1}{2}$ است، مقادیر α، β، γ و λ را محاسبه کنید.

---

۲. (۲۰٪) [نظری و پیاده‌سازی- تحلیل احساسات] هدف از این بخش، شناخت نقاط قوت و ضعف و تاثیر پارامترهای مدل بر دقت بیز ساده و استفاده از آن برای دسته‌بندی نظرهای ثبت شده در اسنپ‌فود[1] است. برچسب هر نظر می‌تواند مثبت (خوشحال یا راضی) و منفی (ناراحت یا ناراضی) است. بایستی احتمال خوشحال بودن نظرهای موجود در دادگان آزمایش را به کمک بیز ساده پیش‌بینی کنید. به این منظور قصد داریم تا با پیاده‌سازی مدل بیز ساده نظرهای مجموعه‌دادگان آزمون را دسته‌بندی کنیم (ما ساختاری را برای شروع پیاده‌سازی در اختیار شما قرار داده‌ایم naïve_bayes.py، لازم است آن را طبق بخش‌های مختلف سوال تکمیل کرده و موارد توضیحی خواسته شده را در گزارش خود ارائه کنید، **دقت کنید که برای آموزش مدل خود به صورت تصادفی ۸۰ درصد از دادگان آموزش را جدا کرده و بر آن‌ها مدل را آموزش دهید**).

۱. **واحدسازی[2]، سبد کلمات[3] و شمارش[4]**: پیش از ساخت مدل باید متن را به نمایشی تبدیل کنیم که برای مدل قابل فهم است. در اینجا از نمایش BOW استفاده می‌کنیم. مدل Naïve Bayes به ترتیب کلمات در سند اهمیت نمی‌دهد و فقط تعداد دفعه‌هایی که کلمه‌ها تکرار می‌شوند مهم است.

[1] Snappfood
[2] Tokenization
[3] Bag-of-Words (BOW)
[4] Counting

h.veisi@ut.ac.ir

به نام خدا

دانشکده سامانه‌های هوشمند

روش‌های یادگیری ماشین در پردازش زبان طبیعی (۸۳۰۴۳۶۸)

نیم‌سال اول ۱۴۰۴-۱۴۰۳

تمرین شماره ۲

تاریخ تحویل:
۱۴۰۳/۰۸/۱۴

۱. تابع tokenize_doc(doc) را پیاده‌سازی کنید. این تابع یک سند (یک رشته) را به عنوان ورودی دریافت می‌کند و آن را به کلمات مختلف جدا می‌کند (از فاصله برای جداسازی استفاده کنید). سپس هر کلمه را به تعداد دفعاتی که در آن سند تکرار شده، توسط یک دیکشنری تصویر کرده و این دیکشنری را به عنوان خروجی باز می‌گرداند. خطوط مرتبط را در فایل برای ارزیابی کد خود از کامنت در بیاورید و اجرا کنید.

۲. تابع update_model را پیاده‌سازی کنید. پیش از شروع، حتما کامنت‌های مرتبط با این تابع را مطالعه کنید، تا بدانید چه چیزی را باید به‌روزرسانی کنید. همچنین به آرگومان‌هایی که کلاس NaiveBayes دریافت می‌کند دقت کنید و در پیاده‌سازی‌های خود در نظر بگیرید. تابع train_model را اجرا کنید. تعداد کلمات موجود در دادگان آموزش چقدر است؟

۳. بیاید به بررسی شمار آمارگان ذخیره شده توسط update_model بپردازیم. به این منظور تابع top_n را پیاده‌سازی کنید. این تابع یک دسته (مقادیر برچسب‌های احساس، برای مثال؛ دسته/کلاس مثبت یا منفی) و یک عدد n را به عنوان ورودی دریافت می‌کند و n کلمه پرتکرار که در سند نسبت به دسته مورد نظر تکرار شده‌اند را به عنوان خروجی باز می‌گرداند. به کمک این تابع، ۱۰ رایج‌ترین کلمه در دسته احساس مثبت و منفی را بدست آورده و در گزارش خود ارائه کنید.

۴. آیا ۱۰ رایج‌ترین کلمه در دسته مثبت/منفی کمکی به تفکیک این دو دسته انجام می‌دهد؟ آیا این فرایند برای دیگر مجموعه‌دادگان در زبان فارسی به نتایج مشابهی منجر می‌شود؟

۲. **احتمال کلمات** و **شبه‌شماره‌ها**[1]: مدل بیز ساده فرض می‌کند که همه ویژگی‌ها با توجه به برچسب کلاس مستقل هستند. در این مساله، این بدان معنی است که احتمال دیدن یک کلمه خاص در یک سند با برچسب کلاس y مستقل از بقیه کلمات در آن سند است.

۱. تابع p_word_given_label را پیاده‌سازی کنید. این تابع باید مقدار $p(w|y)$ را حساب کند (به عبارت دیگر؛ احتمال مشاهده کلمه w با فرض اینکه برچسب سند y است).

۲. به کمک این تابع احتمال کلمه «عالی» و «افتضاح» را برای هر دو برچسب مثبت و منفی حساب کنید. کدام کلمه احتمال بیشتری در دسته مثبت دارد؟ کدام کلمه احتمال بیشتری در دسته منفی دارد؟ آیا این نتایج مطابق انتظار است؟

۳. توضیح دهید که اگر کلمه‌ای در دسته مثبت داده‌های آموزش است ولی در دسته منفی نیست (و حتی بلعکس)، احتمال آن را چگونه محاسبه می‌کنید؟ شرح دهید که چه چیزی مشکل ایجاد می‌کند؟

۴. این مشکل را می‌توان به کمک شبه‌شماره‌ها حل کرد. شبه‌شماره یک مقدار ثابت است که به تعداد تکرار هر کلمه در مدل اضافه می‌شود. از این مقدار برای هموارسازی[2] محاسبه کلماتی استفاده می‌شود که اطلاعات کمی از آن‌ها در دادگان موجود است. به این منظور تابع p_word_given_label_and_pseudocount را

[1] Pseudocounts
[2] Smoothing

h.veisi@ut.ac.ir

به نام خدا

دانشکده سامانه‌های هوشمند

روش‌های یادگیری ماشین در پردازش زبان طبیعی (۸۳۰۴۳۶۸)

نیم‌سال اول ۱۴۰۴-۱۴۰۳

تمرین شماره ۲

تاریخ تحویل:
۱۴۰۳/۰۸/۱۴

پیاده‌سازی کنید.

۳. **احتمال پیشین**[1] و **درست‌نمایی**[2]: همانطور که پیش‌تر گفته شد، مدل بیز ساده فرض می‌کند که همه کلمات در یک سند با توجه به برچست اسناد مستقل از یکدیگر هستند. به همین دلیل می‌توانیم درست‌نمایی یک سند را به صورت زیر تعریف کنیم:

$$p(w_{d_1}, \ldots, w_{d_n} | y_d) = \prod_{i=1}^{n} P(w_{d_i} | y_i)$$

در اینجا w_{d_i}، i امین کلمه در سند d و y_d، برچسب سند d است. با این حال اگر سندی دارای کلمه‌های زیادی باشد، احتمال آن بسیار کوچک می‌شود و با کمبود حسابی[3] مواجه خواهیم شد. هنگام کاربا مدل‌های احتمالی، کمبود حسابی یک مشکل رایج است. اگر با آن آشنا نیستید، می‌توانید یک نمای کلی از ویکی‌پدیا دریافت کنید. یکی از راه‌حل‌های رایج در برخورد با این مشکل، تبدیل فضا به فضای لگاریتمی[4] است.

۱. لگاریتم درست‌نمایی (همانطور که در عبارت بالا آمده است) را حساب کنید.
۲. تابع log_likelihood را پیاده‌سازی کنید (دقت کنید که باید از فراخوانی تابع p_word_given_label_and_pseudocount استفاده کنید).
۳. تابع log-prior را پیاده‌سازی کنید. این تابع یک برچسب را به عنوان ورودی می‌گیرد و لگاریتم قسمتی از دادگان آموزش که این برچسب را دارند، باز می‌گرداند.

۴. **نرمال‌سازی**[5] و **قانون تصمیم‌گیری**[6]: بیز ساده مدلی است که به ما می‌گوید چگونه احتمال پسین یک سند با کلاسی (به عبارت دیگر؛ $p(y_d | w_d)$) را محاسبه کنیم. به کمک قانون بیز داریم که:

$$p(y_d | w_d) = \frac{p(y_d)p(w_d|y_d)}{p(w_d)}$$

در بخش قبل تابع‌های لازم برای محاسبه log prior(log[P(y_d)]) و log likelihood(log[P(w_d|y_d)]) را پیاده‌سازی کرده‌اید. حال لازم است که نرمال‌ساز را محاسبه کنید (P(w_d)).

۱. نرمال‌ساز[7] $p(w_d)$ را بدست آورید.
۲. لگاریتم احتمال پسین را با لگاریتم گرفتن از تساوی بالا، بدست آورید.[8]
۳. یک راه برای دسته‌بندی یک سند این است که لگاریتم احتمال پسین نرمال‌نشده برای هر دو برچسب محاسبه کنیم و سپس argmax بگیریم (به عبارت دیگر؛ برچسبی که منجر به لگاریتم احتمال پیشین نرمال‌نشده‌ی بیشتری می‌شود). لگاریتم احتمال پسین نرمال‌نشده از جمع لگاریتم احتمال پیشین و

[1] Prior
[2] Likelihood
[3] Arithmetic Underflow
[4] Log-Space
[5] Normalize
[6] Decision Rule
[7] Normalizer is denoted as $P(w_d)$, which is the probability of observing the document regardless of its label.
[8] The log of the posterior probability is the logarithm of the expression representing the probability of a certain label given the observed data. The log transformation is often applied to simplify calculations and avoid numerical underflow issues.

h.veisi@ut.ac.ir

لگاریتم درست‌نمایی یک سند بدست می‌آید. چرا به محاسبه لگاریتم نرمال‌ساز نیازی نداریم؟

۴. تابع unnormalized_log_posterior را پیاده‌سازی کنید.

۵. تابع classify را پیاده‌سازی کنید (این تابع باید از لگاریتم نرمال‌نشده احتمال پسین استفاده کند، اما نباید نرمال‌ساز را محاسبه کند).

۵. **ارزیابی**[1]: پس از آموزش مدل و پیاده‌سازی عملکرد دسته‌بندی، قصد داریم که دقت مدل را ارزیابی کنیم.

۱. تابع evaluate_classifier_accuracy را پیاده‌سازی کنید. این تابع باید تمام نمونه‌های موجود در مجموعه‌دادگان آزمایش را برچسب‌گذاری کند. سپس قسمتی از دادگان آزمایش را که درست برچسب‌گذاری شده‌اند را گزارش کنید (شبه‌شماره را برابر با ۱ در نظر بگیرید).

۲. در این بخش با مقادیر مختلف شبه‌شماره (مقادیر؛ [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]) دقت مدل را بررسی کنید. نتایج را به صورت نمودار نمایش دهید (از تابع plot_psuedocounts_vs_accuracy استفاده کنید).

۳. یک نمونه بیابید که دسته‌بند آن را اشتباهی تشخیص داده باشد. فکر می‌کنید که چرا مدل اینگونه عمل کرده است؟ چه راهکارهایی پیشنهاد می‌دهید که مدل این نمونه را درست تشخیص دهد.

۶. **تحلیل کاوشگرانه**[2]: حال از آنجا که مدل ما آموزش دیده است، می‌توان به تحلیل کاوشگرانه داده‌ها پرداخت. دیدیم که ۱۰ رایج‌ترین کلمه[3] برای هر کلاس چندان تبعیض آمیز نیستند. اغلب اوقات، یک آمار توصیفی‌تر، نرخ درست‌نمایی کلمات[4] است. نرخ درست نمایی کلمات را به صورت تعریف می‌کنیم:

$$LR(w) = \frac{P(w\,|y = positive)}{P(w|y = negative)}$$

۱. حال به کمک مدل آماده شده، به با توجه به تعریف، اگر LR برای کلمه‌ای برابر با ۵ بدست آید، به این معنی است که احتمال تعلق داشتن آن کلمه به دسته مثبت ۵ برابر محتمل‌تر از تعلق داشتن به دسته منفی است. همچنین اگر LR برای کلمه‌ای برابر با ۰.۳۳ بدست آید، به این معنی است که احتمال تعلق داشتن آن کلمه به دسته مثبت یک سوم احتمال تعلق آن به دسته منفی است.

۲. خروجی تابع LR در چه بازه‌ای است؟

۳. تابع likelihood_ratio را پیاده‌سازی کنید (این تابع یک کلمه را به عنوان ورودی می‌گیرد و نرخ درست‌نمایی کلمات را مطابق فرمول بالا محاسبه می‌کند).

۴. LR را برای دو کلمه «عالی» و «افتضاح» حساب کنید. مقادیر بدست آمده را با نرخ ۱۰ رایج‌ترین کلمه که در بخش‌های قبل بدست آوردید، مقایسه کنید.

۵. توضیح دهید که چگونه مقادیر LR کلمات به مدل بیز ساده مرتبط هستند؟ برای مثال اگر LR کلمه‌ای

[1] Evaluation
[2] Exploratory Analysis
[3] Top 10 Most Common Words
[4] Words Likelihood Ratio (LR)

h.veisi@ut.ac.ir

۱ باشد، آیا به این معنی است این کلمه برای دسته‌بند بیز ساده با اهمیت است یا خیر؟ اگر LR برای کلمه‌ای بسیار با مقدار ۱ اختلاف داشته باشد (برای مثال؛ ۰.۰۱ و ۱۰۰)، آیا به این معناست که این کلمه برای دسته‌بند بیز ساده با اهمیت/بی اهمیت است؟ همچنین بیان کنید که کلمه‌ای که دارای LR=0.01 است، در مقایسه با کلمه‌ای که LR=100 دارد، در دسته‌بندی چه تاثیری دارد؟ (موارد خواسته شده را به صورت دقیق توضیح دهید)

۷. **تعمیم به چند کلاس**: در دنیای واقعی مساله‌های دسته‌بندی چند کلاسه[1] نسبت به دسته‌بندی دو کلاسه به مراتبط بیشتر هستند.

۱. اگر قصد داشته باشیم مدل حال حاضر را به منظور دسته‌بندی چند کلاسه تغییر دهیم، آمار شمارشی[2] که ذخیره می‌کنیم چگونه تغییر می‌کند؟ تمام نمونه‌های موجود در مجموعه‌دادگان آزمایش را دسته‌بندی کنید و بخشی از نمونه‌هایی را که به درستی دسته‌بندی شده‌اند را در گزارش خود ارائه نمایید. دقت دسته‌بندهای خود را با شبه‌شماره برابر با مقدار ۱، گزارش کنید.

۲. نرمال‌ساز چگونه تغییر می‌کند؟

۳. قانون تصمیم‌گیری جدید چه خواهد بود (به عبارت دیگر؛ تابع دسته‌بندی چگونه تغییر می‌کند)؟

۸. **ارزیابی مجموعه‌دادگان آزمایش**: دادگان آزمایش test.csv در اختیار به شما قرار گرفته است، لازم است که شما به کمک مدل بیز ساده که تکمیل کردید به دادگان تست برچسب بزنید. قصد داریم به میزان خوب بودن مدل شما امتیازی بدهیم، امتیاز نهایی مدل شما تابعی از سطح زیر نمودار منحنی مشخصه عملکرد سیستم است. AUC-ROC مشخصه‌ای از میزان تطابق بین برچسب‌های داده‌شده و برچسب‌های پیش‌بینی‌شده است. امتیاز نهایی مدل به کمک رابطه زیر محاسبه می‌شود.

$$\text{score} = 4 \times (\text{AUC-ROC} \times 100 - 50)$$

---

۳. (۴۰٪) [**پیاده‌سازی- سخنان تنفرپراکنانه**] هدف از این بخش تمرین سه روش Naïve Bayes، Logistic Regression و KNN است. دادگان سخنان تنفرآمیز[3] در اختیار شما قرار گرفته است. قصد داریم به کمک الگوریتم‌های یادگیری ماشین، از دادگان آموزش برای یادگیری مدل و دادگان آزمایش برای بررسی وضعیت یادگیری مدل استفاده کنیم. اگر سخنی دارای سه ویژگی زیر باشد، به عنوان سخن تنفرآمیز در نظر گرفته می‌شود:

- حمله عمدی[4] باشد.
- به سمت یک گروه خاصی از افراد باشد.
- توسط ویژگی‌های آن گروه منشاء بگیرد.

[1] Multi-Class Classification
[2] Count Statistics
[3] Hatespeech
[4] Deliberate attack

h.veisi@ut.ac.ir

به نام خدا

دانشکده سامانه‌های هوشمند

روش‌های یادگیری ماشین در پردازش زبان طبیعی (۸۳۰۴۳۶۸)

نیم‌سال اول ۱۴۰۴-۱۴۰۳

تمرین شماره ۲

تاریخ تحویل:
۱۴۰۳/۰۸/۱۴

برچسب هر عبارت با مقدار ۱ (به معنای تنفرآمیز بودن) و ۰ (به معنای تنفرآمیز نبودن) نشان‌داده می‌شود.

ساختار پیشنهادی برای طراحی در قالب فایل‌های utils.py، classifier.py و main.py در اختیار شما قرار گرفته است. برای پیاده‌سازی دسته‌بندهای Naïve Bayes، Logistic Regression و KNN شما نیاز به پیاده‌سازی الگوریتم‌های خواسته شده در uitils.py و classifier.py دارید. توضیح‌های مورد نیاز برای هر پیاده‌سازی در ادامه برای شما ارائه می‌شود.

۱. **استخراج ویژگی**: در Utils.py تابع tokenize عمل جداسازی کلمه‌ها با الگوی داده شده انجام می‌شود. برای مثال جمله "I love nlp" را به لیستی از کلمه‌های ["I", "love", "nlp"] تبدیل می‌کند. به کمک این تابع می‌خواهیم استخراج ویژگی‌های یک‌تایی‌ها[1] و دوتایی‌ها[2] را پیاده‌سازی کنیم. برای هر «استخراج‌کننده ویژگی» چهار تابع init ، fit، transform و transform_list تعریف شده است که لازم است آن‌ها را تکمیل کنید. در ادامه، تابع‌های اشاره شده برای روش Unigram توضیح داده می‌شود.

__init__(self)

در این تابع یک دیکشنری به نام unigram را مقداردهی اولیه می‌کنید.

fit(self, text_set: list)

در این قسمت لازم است تا دیکشنری unigram را به کمک text_set، که لیستی از جمله‌های جداسازی شده است، مقداردهی کنید. برای مثال اگر:

```
Text_list = [["I", "love", "nlp"], ["I", "like", "python"]]
```

در این صورت پس از اجرای تابع بر روی text_list و self.unigram، نتیجه به صورت زیر خواهد بود:

```
{"I": 0, "love": 1, "nlp": 2, "like": 3, "python": 4}
```

توجه داشته باشید که اندازه حروف[3] اهمیتی ندارد، برای مثال دو کلمه "Python" و "pYthON" یک کلمه در نظر گرفته می‌شوند.

transform(self, text: list)

در این تابع به کمک دیکشنری بدست آمده از بخش قبل، یک بردار ویژگی برای جمله text بدست آورید. برای مثال اگر:

```
Text = ["I", "love", "nlp"]
```

در این صورت پس از اجرای تابع بر روی text، بردار ویژگی زیر را به عنوان خروجی بازمی‌گردانیم:

```
[1, 1, 1, 0, 0]
```

transform_list(self, text_set: list)

برای لیستی از جمله‌های جداسازی شده، لیستی از بردارهای ویژگی را به عنوان خروجی بازگردانید. برای مثال:

```
Text_list = [["I", "love", "nlp"], ["I", "like", "python"]]
```

در این صورت خروجی این تابع باید یک آرایه NumPy به شکل زیر باشد:

[1] UnigramFeature
[2] BigramFeature
[3] Lowercase or Uppercase

h.veisi@ut.ac.ir

```
array([[1, 1, 1, 0, 0], [1, 0, 0, 1, 1]])
```

توضیحات داده شده برای هر چهار تابع را در کلاس UnigramFeature تکمیل کنید و به کمک این توابع یک استخراج‌کننده bigram در کلاس BigramFeature نیز بنویسید.

۲. **پیاده‌سازی دسته‌بندهای** Naïve Bayes، Logistic Regression و KNN: در Classifier.py دو کلاس NaiveBayesClassifier و LogisticRegressionClassifier و KNNClassifier داده شده است، برای هر الگوریتم لازم است که سه تابع init، fit و predict را تکمیل کنید. در ادامه جزئیات تابع‌های گفته شده برای دسته‌بند Naïve Bayes ارائه می‌شود:

__init__(self)

دسته‌بند Naïve Bayes از احتمال هر دسته و احتمال ویژگی‌ها برای هر دسته برای دسته‌بندی استفاده می‌کند، پس دو دیکشنری برای ذخیره این احتمال‌ها تعریف کنید.

fit(self, X, Y)

در این قسمت باید احتمال‌های تعریف شده را برای دسته‌ها و ویژگی‌ها برای داده X و برچسب y حساب کنید (از laplace smoothing استفاده کنید).

predict(self, X)

برای ارزیابی مدل بدست آمده، تابعی بنویسید که دادگان آزمایش را توسط مدلی که آموزش داده‌اید، برچسب‌گذاری کند (خروجی این تابع باید یک آرایه NumPy است).

توابع گفته شده را در کلاس NaiveBayesClassifier کامل کنید. سپس همین کار را برای مدل LogisticRegression در کلاس LogisticRegressionClassifier انجام دهید.

در رگرسیون لاجیستیک بردار وزنی به اندازه تعداد ویژگی‌ها تعریف کنید. سپس این وزن‌ها را در تابع fit به کمک فرمول‌های زیر، بروزرسانی کنید.

محاسبه تابع هزینه:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^i * log\left(h\theta(x^i)\right) + (1 - y^i) * log(1 - h\theta(x^i)\right]$$

محاسبه گرادیان:

$$\frac{\partial J(\theta)}{\partial \theta j} = \frac{1}{m}\sum_{i=1}^{m}(h\theta(x^i) - y^i) * x_j^i$$

بروزرسانی وزن‌ها به کمک گرادیان:

$$\theta j = \theta j - \alpha * \frac{\partial J(\theta)}{\partial \theta j}$$

به صورت پیش‌فرض از نرخ یادگیری 0.01 و تعداد ایپاک 100 استفاده کنید و به کمک وزن‌های بدست آمده، دادگان آزمایش را پیش‌بینی کنید.

$$Predict = \sigma(XW)$$

h.veisi@ut.ac.ir

به نام خدا

۳. **اجرای مدل‌ها و ارزیابی نتایج**: در این قسمت دقت مدل‌های ساخته شده را بررسی می‌کنیم. به این منظور، خط فرمان زیر را در محیط Colab اجرا کنید (اگر از cmd/shell/bash استفاده می‌کنید، به جای !python از python استفاده کنید).

```
!python main.py -model AlwaysPredictZero
```

در صورت موفق بودن اجرای این فرمان، خروجی به صورت زیر خواهد بود.

```
Namespace(model='AlwaysPredictZero', feature='unigram', path='./data/')
==== Train Accuracy ====
Accuracy : 956 / 1913 = 0.4997
```

جهت ارزیابی مدل NaiveBayesClassifier با استخراج‌کننده ویژگی یک‌تایی، دستور زیر را اجرا کنید:

```
!python main.py -model NaiveBayes -feature unigram
```

دقت بدست آمده برای مدل‌های Naïve Bayes، Logistic Regression و KNN به کمک دو روش استخراج ویژگی یک‌تایی و دوتایی در جدول زیر گزارش کنید.

جدول ۱: نتایج مدل‌های بدست آمده

استخراج ویژگی / مدل	استخراج‌کننده ویژگی یک‌تایی	استخراج‌کننده ویژگی دوتایی
بیز ساده		
رگرسیون لاجستیک		
k-نزدیک‌ترین همسایه		

۴. (۲۵٪) **[پیاده‌سازی: برچسب‌زنی اجزای کلام با درخت تصمیم]** هدف این تمرین پیاده‌سازی برچسب‌زن اجزای کلام برای زبان فارسی با استفاده از روش درخت تصمیم است. در این تمرین از داده‌های ارائه شده به همراه تمرین استفاده کنید. داده‌های داده شده شامل یک فایل آموزش با اسم POST-Persian-Corpus-Train.txt (حدود ۳۰۰۰ جمله) و یک فایل آزمون با اسم POST-Persian-Corpus-Test.txt (حدود ۵۵۰ جمله) است که تمام کلمات آن برچسب‌گذاری دستی (با تعداد ۲۱ برچسب) شده‌اند. برای ساخت درخت از داده فایل آموزش و برای ارزیابی کار خود و تست کردن کیفیت سیستم ساخته شده، از داده فایل آزمون استفاده شود. ویژگی‌های مورد استفاده برای حل این سوال را بررسی کرده و در پاسخ خود بیان کنید که از چه ویژگی‌هایی استفاده کرده‌اید.

برای فاز آزمون، از معیار F1 استفاده کنید و همچنین ماتریس Confusion را رسم کنید.

h.veisi@ut.ac.ir