Programska dokumentacija

Domagoj Bošnjak, Iskra Gašparić

1 Podaci

Baza podataka korištena za testiranje algoritama pohranjena je u datoteci *spam.csv*. Podatke iz datoteke spremamo u tip *pandas dataframe*. Npr.

	Class	Content	Class_number
0	ham	Go until jurong point, crazy.. Available only ...	0
1	ham	Ok lar... Joking wif u oni...	0
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	1
3	ham	U dun say so early hor... U c already then say...	0
4	ham	Nah I don't think he goes to usf, he lives aro...	0

Potrebni input za bilježnice:

- **kmeans_knn.ipynb**
  spam.csv - dataset SMS poruka
- **nn.ipynb**
  spam.csv - dataset SMS poruka
  X_trainTokenized.csv,
  X_testTokenized.csv,
  ShuffledLabels.csv - tokenizirane datoteke dobivene kao output u bilježnici *kmeans_knn.ipynb*

U repozitorij su priloženi primjeri posljednje tri datoteke kako ih se ne bi moralo ručno premještati.

2 K-means i k-NN algoritmi

Sam kod nalazi se u bilježnici *kmeans_knn.ipynb*. Kod je podijeljen u sekcije pa je tako i dokumentiran.

- **Tokenization**
  Da bismo klasificirali SMS poruke moramo ih moći usporediti pa radimo tzv. *tokenizaciju*. Prvo se svaka poruka predprocesira micanjem interpunkcijskih znakova, stop riječi (the, at, ...) te promjenom svih slova u

mala slova. Zatim koristimo *Porterov algoritam* za pretvaranje riječi u korijene riječi(tokenizacija). Algoritam je implementiran u *nltk* paketu.

Zatim, umjesto da pamtimo od kojih riječi (odnosno korijena riječi - *tokena*) je svaki SMS sastavljen, radimo varijablu *vocabulary*, tipa riječnik takva da svakom jedinstvenom tokenu pridruži jedinstven broj. Zatim u varijabli *finalDocumentIndices* SMS poruke pamtimo u obliku liste brojeva, koji odgovaraju pripadnim tokenima iz SMS poruka.

Funkcija *tokenizationWithVocabulary* radi esencijalno istu stvar, samo više ne tražimo riječi iz SMS poruka nego koristimo već postojeći riječnik da bismo klasificirali dokumente. Ova funkcija koristi se za karakterizaciju testnog skupa.

- **Feature extraction and TFIDF matrix construction**
  Za značajke (*features*) na osnovu kojih uspoređujemo poruke odabrali smo broj pojava globalno najčešćih riječi. Prvo pronađemo sve riječi koje imaju barem *frequencyThreshold* pojavljivanja te ih spremamo kao *mostFrequentWords*. Taj objekt analogno kao ranije pretvaramo u riječnik.

  Funkcijom *createTfidfMatrix* gradimo matricu značajki (*feature matrix*). Koristimo gotovu funkciju iz *sklearn* paketa. Ta se matrica koristi za daljnje algoritme.

- **K-means**
  Funkcija je standardni *K-means* algoritam te za input uzima podatke za trening, podatke za testiranje te pripadne stvarne spam i ham vrijednosti. Koristi se $k = 2$ te funkcija printa koliko spam, odnosno ham SMS poruka jest u svakoj klasi.

- **K-nearest neighbors**
  Potpuno analogno funkciji *K-means*, osim što koristimo *k-nn* algoritam, također dan u *sklearn* paketu.

- **Testing**
  Prije samog testiranja jednostavnim algoritmom randomiziramo poredak podataka i oznaka. Redom pozivamo sve funkcije te pripadne algoritme. Konačno,

3 Neuronska mreža

Kod se nalazi u bilježnici *nn.ipynb*. Kao i u prošlom poglavlju, dokumentiramo sekcije.

- **Reading and preprocessing data**
  Učitavamo bazu iz datoteke *spam.csv*. Poruke se pretvore u nizove integera koji su brojevna reprezentacija pojedinog znaka. Nakon skaliranja vrijednosti na interval $[0, 1]$ i rezanja poruka na istu duljinu, dijelimo podatke u dva skupa - training i testing skup.

- **Sigmoid function, error function and intializing parameters**
  Kao funkciju aktivacije odabrali smo sigmoidalnu funkciju $1/(1 + e^{-x})$. Računa se i funkcija greške.

- **Gradient descent**
  Uz unaprijed zadan broj iteracija i parametar učenja, radi se gradijentni spust i ažuriraju se težine iz modela.

- **Prediction**
  Izvrednjavanjem sigmoidalne funkcije, za zadani input neuronska mreža procjenjuje je li SMS ham ili spam (ovisno je li vrijednost funkcije u intervalu $[0, 0.5\rangle$ ili $[0.5, 1]$). Funkcija može raditi na više podataka istovremeno.

- **Complete network**
  Funkcija *neuralNetwork* spaja sve prethodno definirane funkcije u cjelinu. Učenje neuronske mreže testira se na training i testing setu. Također se ispituje točnost uspoređivanjem rezultata iz algoritma sa stvarnim vrijednostima te se ona ispisuje.

- **Testing the network**
  Ispitujemo efikasnost dva slučaja. Najprije učimo neuronsku mrežu pomoću podataka predprocesiranih u prvoj sekciji (pretvaranje znakova u integere). Zatim se testiraju rezultati nastali na temelju analize ključnih riječi (kao u algoritmu k-sredina).