Feature Engineering

Natural Language Processing

University of Maryland

Discovery

Feature Engineering

In the `data` directory for `nlp-hw` repo, there are files called `inclass_feateng_train.csv` (and likewise for dev, test).

Load Packages

```
import pandas as pd
import numpy as np
import sklearn
from sklearn.utils.validation import check_is_fitted
from sklearn.exceptions import NotFittedError
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction import DictVectorizer
```

Create function to generate features

```python
def extract_features(sentence_list):
    d = {}
    d["length"] = len(sentence_list)
    return d
```

Load Data

```
vec = DictVectorizer()
X = {}
Y = {}
df = {}
for fold in ["train", "dev", "test"]:
    df[fold] = pd.read_csv("%s.csv" % fold)
    Y[fold] = df[fold]["label"]
    try:
        check_is_fitted(vec)
        X[fold] = vec.transform(extract_features(x.split())
                                for x in df[fold]['text'])
    except NotFittedError as exc:
        X[fold] = vec.fit_transform(extract_features(x.split())
                                    for x in df[fold]['text'])
```

Train Classifier

```
classifier = LogisticRegression()
classifier.fit(X["train"], Y["train"])
```

Look at Features, Compute Accuracy

```
coef = pd.DataFrame(zip(vectorizer.feature_names_,
                        np.transpose(classifier.coef_)),
                    columns=['features', 'coef'])
coef.loc[len(coef.index)] = ['Intercept',
                             classifier.intercept_]
print(coef.head())

print("Accuracy: %f" %
      classifier.score(X["dev"], Y["dev"]))
```

Results

```
    features                          coef
0     length  [0.009380652536927882]
1  Intercept  [-0.1161938266865230 7]
Accuracy: 0.485398
```

Results

```
    features                      coef
0     length  [0.009380652536927882]
1  Intercept  [-0.1161938266865230 7]
Accuracy: 0.485398
```

Not great! (Roughly balanced dataset)

Look at Errors

```python
df["dev"]["error"] = (df["dev"]["label"].astype(int) -
                      classifier.predict_proba(X["dev"])[:,1])
df["dev"].sort_values(by='error', key=abs,
                      ascending=False, inplace=True)
print(df["dev"].head())
num_rows = 0
for index, row in df["dev"].iterrows():
    num_rows += 1
    print(row["error"], len(row["text"].split()),
          row["text"])
    if num_rows > 10:
        break
```

```
-0.7005699501953168 103 I'm sending you a couple of custom
-0.651327096225724 79 Two millions were added to what had
-0.6384386405287599 73 Like Eliot , in my fantasies , I ha
-0.6362704602163262 72 So , for happy years , Helva scoote
-0.6297329173061533 69 When he was bent over behind the wh
```

Notice anything?

- The examples are long, sure, but is there anyting suspicious about their <u>exact</u> length?
- Add in new feature and see if it helps...
- Remember that you need to redo `fit_transform`

More hints

- Can perfectly reconstruct with four features
- Two features you can figure out by using all words as features and seeing patterns on the top words
- Pay attention to how words are written
- Pay attention to what words mean
- Pay attention to the order of words

Even more hints

1. Feature based on length has positive weight, but isn't monotonic. . . but is monotonic on the length of something related to the length. Do this feature first.
2. Feature based on how words are written has a positive weight and is easy to compute.
3. Feature based on what words mean has a negative weight and will require using a dictionary to get it perfectly right. Consider using `wordnet` from `nltk`
4. The value of the feature based on the order of words is around 0.5 for most sentences and ranges between 0.0 and 0.1, and the weight of the feature is negative. I'd recommend looking for this feature last.

Exam Question

You have a logistic regression classifer that takes in words and then classifies them. What three features would explain the following inputs and probabilities?

Word	$p(y \mid x)$
short	0.11
hot	0.26
saving	0.26
taking	0.5
surely	0.73
bigly	0.88

Solution

- Intercept: -1
- Starts with “s”: -1
- Ends with “ing”: +1
- Ends with “ly”: +3