Homework 5: Working With Text Data and Uncertainty

Due 23:59 on Tuesday 13 December 2022

You will do this assignment individually and submit your answers via the Blackboard course website. The goal of this homework is to gain experience with text data and simple uncertainty estimation.

1. Problem [49pts]

The code for this problem follows the code from the 11/22/22 lecture's "Working with text data" section.

Load some sample text data using sklearn:

```
import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import fetch_20newsgroups
twenty_train = fetch_20newsgroups(subset='train',
                                  categories=['alt.atheism',
                                              'soc.religion.christian',
                                              'comp.graphics'],
                                  shuffle=True, random_state=42)
twenty_test = fetch_20newsgroups(subset='test',
                                 categories=['alt.atheism',
                                             'soc.religion.christian',
                                             'comp.graphics'],
                                 shuffle=True, random_state=42)

# This is out of distribution data. Not seen in training.
twenty_ood = fetch_20newsgroups(subset='test',
                                categories=['sci.med',
                                            'sci.space',
                                            'rec.autos'],
                                shuffle=True, random_state=42)
```

Print an example data point:

```
print(twenty_train.data[0])
```

Turn the text data into a set of numerical vectors, using count vectorization. And use TFIDF to standardize the counts between documents:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
```

```
count_vect = CountVectorizer(stop_words='english')
X_train_counts = count_vect.fit_transform(twenty_train.data)
X_test_counts = count_vect.transform(twenty_test.data)
X_ood_counts = count_vect.transform(twenty_ood.data)

tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
X_test_tfidf = tfidf_transformer.transform(X_test_counts)
X_ood_tfidf = tfidf_transformer.transform(X_ood_counts)
```

Next train multiple neural network models. The multiple models will allow us to quantify uncertainty in our predictions. For more background, see the following two papers if you are interested.

1. https://proceedings.neurips.cc/paper/2017/file/9ef2ed4b7fd2c810847ffa5fa85bce38-Paper.pdf
2. https://arxiv.org/pdf/1803.08533.pdf

```
from sklearn.neural_network import MLPClassifier

models = [MLPClassifier(hidden_layer_sizes=(50,50)),
          MLPClassifier(hidden_layer_sizes=(50,50)),
          MLPClassifier(hidden_layer_sizes=(50,50)),
          MLPClassifier(hidden_layer_sizes=(50,50)),
          MLPClassifier(hidden_layer_sizes=(50,50))]

for model in models:
    model.fit(X_train_tfidf, twenty_train.target)
```

Now, sample softmax outputs for each of the models. We will track in distribution and out of distribution test data separately.

```
# Predict on test and out of distribution data.
preds_test = []
preds_ood = []
for model in models:
    preds_test.append(model.predict_proba(X_test_tfidf))
    preds_ood.append(model.predict_proba(X_ood_tfidf))

# The shape is (num models, num data points, num classes).
preds_test = np.array(preds_test)
preds_ood = np.array(preds_ood)
```

Next, evaluate the model ensemble on the test data.

```
from sklearn.metrics import classification_report
print(classification_report(twenty_test.target,
np.argmax(np.mean(preds_test,axis=0),axis=1)))
```

Next measure the uncertainty using the mean (across classes) softmax variance (across models). We can then compare uncertainty for in and out of distribution data.

```
indist_uncertainty = np.mean(np.var(preds_test,axis=0),axis=1)
ood_uncertainty = np.mean(np.var(preds_ood,axis=0),axis=1)
```

Show that out of distribution data tend to have higher uncertainty.

```
plt.figure(figsize=(14,7))
plt.hist(indist_uncertainty, label='in distribution',
alpha=0.4, density=True, bins=20, range=(0,0.006))
plt.hist(ood_uncertainty, label='out of distribution',
alpha=0.4, density=True, bins=20, range=(0,0.006))
plt.legend()
plt.xlabel('Uncertainty')
plt.ylabel('Density')
plt.title('Uncertainty')
```

Finally, show that wrong predictions tend to have higher uncertainty.

```
is_correct = twenty_test.target==np.argmax(
    np.mean(preds_test,axis=0),axis=1)

# We can compare uncertainty for test data points
# the model got wrong and correct.
correct_uncertainty = np.mean(np.var(
    preds_test[:,is_correct,:],axis=0),axis=1)
mistake_uncertainty = np.mean(np.var(
    preds_test[:,~is_correct,:],axis=0),axis=1)

# Wrong predictions tend to have higher uncertainty.
plt.figure(figsize=(14,7))
plt.hist(correct_uncertainty, label='correct prediction',
alpha=0.4, density=True, bins=20, range=(0,0.006))
plt.hist(mistake_uncertainty, label='mistake',
alpha=0.4, density=True, bins=20, range=(0,0.006))
plt.legend()
plt.xlabel('Uncertainty')
plt.ylabel('Density')
plt.title('Uncertainty')
```

Conclude by explaining in words why uncertainty estimation is valuable.

2. Calibration [1pt]

Approximately how long did this homework take you to complete?