2-dimensionality-reduction

November 22, 2018

0.1 Dimensionality Reduction

This notebook investigate whether the data preprocessing technique dimensionality reduction can affect the final classification performance.

```
In [1]: import sys, argparse
        sys.path.append('..')
        import helper
        import numpy as np
        import matplotlib.pyplot as plt

        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.svm import LinearSVC
        from sklearn import metrics
        from sklearn.decomposition import PCA
        from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

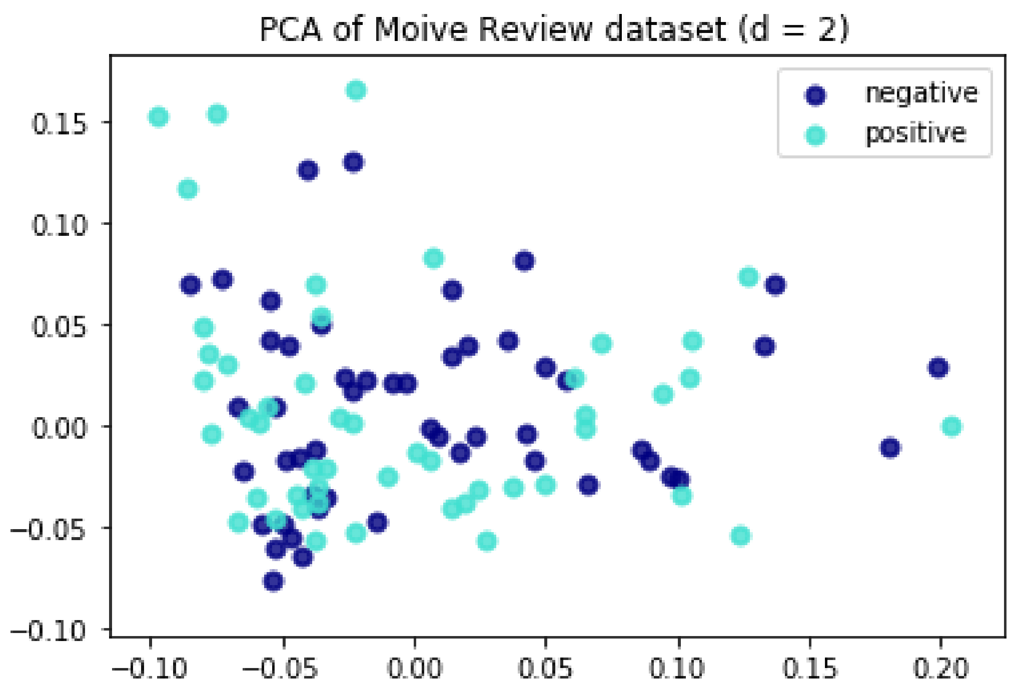
0.1.1 1. Data Preprocessing

```
In [2]: pos_examples = [s.decode("utf-8", "ignore").strip() for s in list(open(helper.mr_pos_da
        neg_examples = [s.decode("utf-8", "ignore").strip() for s in list(open(helper.mr_neg_da
        pos_nums, neg_nums = len(pos_examples), len(neg_examples)
        x = pos_examples + neg_examples
        x = [helper.clean_str(sentence) for sentence in x]
        pos_labels = [1 for _ in range(pos_nums)]
        neg_labels = [0 for _ in range(neg_nums)]
        y = pos_labels + neg_labels
        x, y = np.array(x), np.array(y)
        x_train, y_train, x_dev, y_dev = helper.split_train_dev(x, y)

        # TF-IDF
        tfidf = TfidfVectorizer(min_df=2, ngram_range=(1,2))
        tfidf.fit(x_train)
        x_train_tf = tfidf.transform(x_train).toarray()
        x_dev_tf = tfidf.transform(x_dev).toarray()

        # PCA
```

```
pca = PCA(n_components=2)
x_train_pca = pca.fit(x_train_tf).transform(x_train_tf)
```

0.1.2 2. Reducetion 2-dim

```
In [3]: x_pca, y_pca = x_train_pca[:100], y_train[:100]
        plt.figure()
        colors = ['navy', 'turquoise']
        labels = ['negative', 'positive']

        for color, i, label in zip(colors, [0, 1], labels):
            plt.scatter(x_pca[y_pca == i, 0], x_pca[y_pca == i, 1], color=color, alpha=.8, lw=
            plt.legend(loc='best', shadow=False, scatterpoints=1)
            plt.title('PCA of Moive Review dataset (d = 2)')
```



0.1.3 3. Can dimensionality reduction technique affect the classification performance?

We use the best performance classify linear svc as the baseline in this experiment. In this case, the accuracy drop with the decrement of size of features, interpreting that the dimensionality reduction may lose the information of the original dataset.

```
In [4]: dims = [1000, 500, 100]
        for d in dims:
            tfidf = TfidfVectorizer(min_df=2, ngram_range=(1,2))
            tfidf.fit(x)
```

```
    x_tf = tfidf.transform(x).toarray()
    pca = PCA(n_components=d)
    x_pca = pca.fit(x_tf).transform(x_tf)
    x_train_tf, y_train, x_dev_tf, y_dev = helper.split_train_dev(x_pca, y)
    svc = LinearSVC()
    svc.fit(x_train_tf, y_train)
    predicted = svc.predict(x_dev_tf)
    print("Accuracy: {0:.4f}".format(metrics.accuracy_score(predicted, y_dev)))
    print(metrics.classification_report(predicted, y_dev))
```

```
Accuracy: 0.7655
             precision    recall  f1-score   support

          0       0.76      0.77      0.77       535
          1       0.77      0.76      0.76       531

avg / total       0.77      0.77      0.77      1066

Accuracy: 0.7439
             precision    recall  f1-score   support

          0       0.74      0.75      0.75       532
          1       0.75      0.74      0.74       534

avg / total       0.74      0.74      0.74      1066

Accuracy: 0.6829
             precision    recall  f1-score   support

          0       0.64      0.70      0.67       491
          1       0.72      0.66      0.69       575

avg / total       0.69      0.68      0.68      1066
```