cme250_hw3_solutions

February 25, 2019

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        %matplotlib inline

        from sklearn.svm import SVC
        from sklearn.preprocessing import Imputer
        from sklearn.model_selection import GridSearchCV, cross_val_score, KFold, train_test_split
        from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix
```

1 Question 1. Young People Survey

1.1 (a) Mean Imputation

```
In [2]: # use pandas to read in .csv file
        path = '../data/hw3.csv'
        raw = pd.read_csv(path)
```

```
In [3]: X = raw.drop('Gender', axis=1)
        y = raw['Gender']
```

```
In [4]: # perform mean imputation
        imp = Imputer(strategy='mean')
        imp.fit(X)
        X_imp = imp.transform(X)
```

```
In [5]: # make X a dataframe again
        X_imp = pd.DataFrame(X_imp, columns=X.columns)
```

1.2 (b) Effect of Mean Imputation on Variance

```
In [6]: # original dataset standard deviation of height variable
        print("Standard deviation before mean imputation: {:0.4f}".format(
                np.nanstd(X['Height']), ddof=1))
```

```
Standard deviation before mean imputation: 10.0403
```

```
In [7]: # mean-imputed dataset standard deviation of height variable
        print("Standard deviation after mean imputation: {:0.4f}".format(
                np.nanstd(X_imp['Height']), ddof=1))
```

```
Standard deviation after mean imputation: 9.9421
```

The sample standard deviation decreased after mean imputation. This happened because sample standard deviation is a measure of the distance between sample values and the mean value in the dataset. Its formula is

$s_n = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$

By performing mean imputation, we increased n but did not increase $\sum_{i=1}^{n}(x_i - \bar{x})^2$. Therefore s_n decreased.

1.3 (c) Non-Nested Cross-validation, SVM

```
In [8]: # define a support vector machine classifier and values of gamma to search over
        svm = SVC()
        gammas = np.logspace(-6, 0, 7)
        params = {'gamma': gammas}
```

```
In [9]: print(params)
```

```
{'gamma': array([1.e-06, 1.e-05, 1.e-04, 1.e-03, 1.e-02, 1.e-01, 1.e+00])}
```

```
In [10]: # split 20% of data into test set
         X_trainval, X_test, y_trainval, y_test = train_test_split(
             X_imp, y, test_size=0.2, random_state=0)
```

```
In [11]: # define k-fold splits and grid search cross-validation objects
         inner_cv = KFold(n_splits=5, shuffle=True, random_state=0)
         gridcv = GridSearchCV(estimator=svm, param_grid=params, cv=inner_cv)
         gridcv.fit(X_trainval, y_trainval)
```

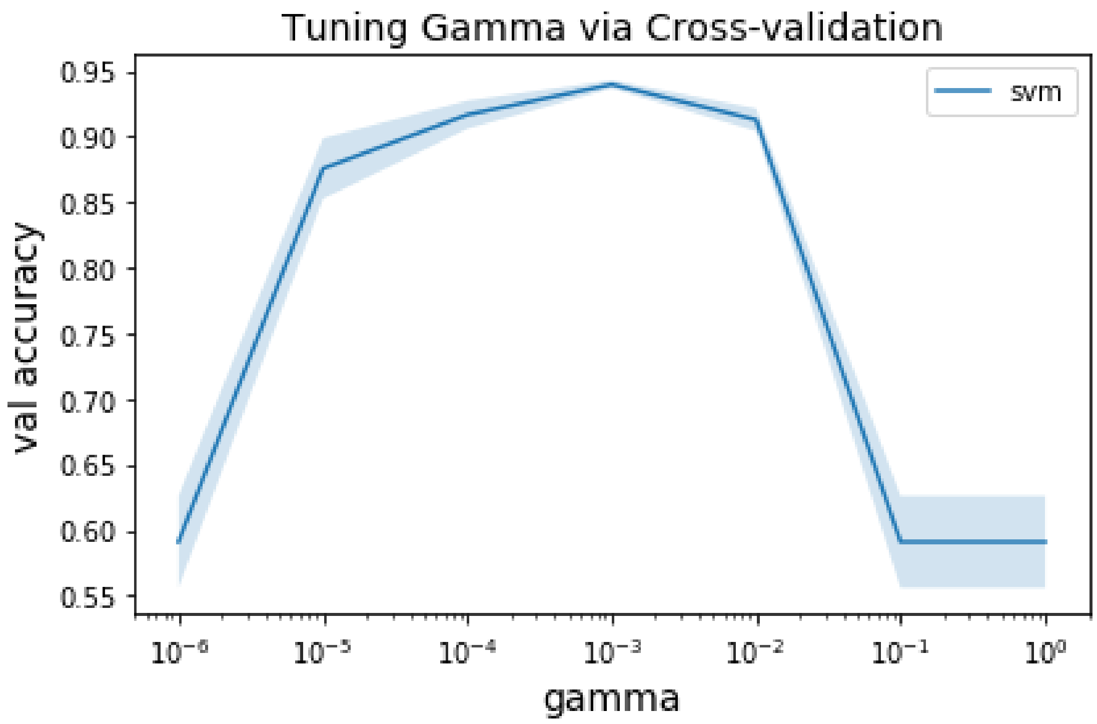
```
Out[11]: GridSearchCV(cv=KFold(n_splits=5, random_state=0, shuffle=True),
                error_score='raise',
                estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
           decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
           max_iter=-1, probability=False, random_state=None, shrinking=True,
           tol=0.001, verbose=False),
                fit_params=None, iid=True, n_jobs=1,
                param_grid={'gamma': array([1.e-06, 1.e-05, 1.e-04, 1.e-03, 1.e-02, 1.e-01, 1.e+00])},
                pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
                scoring=None, verbose=0)
```

```
In [12]: # accuracy values for each hyperparameter combination
         scores = gridcv.cv_results_['mean_test_score']
```

```
In [13]: # standard deviation of accuracy values for each hyperparameter combination
         score_sds = gridcv.cv_results_['std_test_score']
```

```
In [14]: # plot accuracy scores vs. gammas
         plt.semilogx(gammas, scores)
         plt.fill_between(gammas, scores-score_sds,
                          scores+score_sds, alpha=0.2)

         # plot title, axis labels, etc.
         fs = 14
         plt.title('Tuning Gamma via Cross-validation', fontsize=fs)
         plt.xlabel('gamma', fontsize=fs)
         plt.ylabel('val accuracy', fontsize=fs)
         plt.legend(['svm'])
         plt.tight_layout()
         plt.show()
```



```
In [15]: # best hyperparameters
         gridcv.best_params_

Out[15]: {'gamma': 0.001}
```

1.4 (d) Generalization Error

```
In [16]: svm = SVC(gamma=0.001)
         svm.fit(X_trainval, y_trainval)
         y_pred = svm.predict(X_test)

In [17]: print("Test set accuracy: {:0.4f}".format(accuracy_score(y_test, y_pred)))

Test set accuracy: 0.9184

In [ ]:
```