Gr?ficas fourier nystroem

June 14, 2018

```
In [1]: %matplotlib inline
        import matplotlib.pyplot as plt
        import numpy as np
        from time import time

        # Import datasets, classifiers and performance metrics
        from sklearn import datasets, svm, pipeline
        from sklearn.kernel_approximation import (RBFSampler,
                                                  Nystroem)
        from sklearn.decomposition import PCA

        from sklearn.tree import DecisionTreeClassifier

In [2]: # The digits dataset
        digits = datasets.load_digits(n_class=9)

In [3]: # To apply an classifier on this data, we need to flatten the image, to
        # turn the data in a (samples, feature) matrix:
        n_samples = len(digits.data)
        data = digits.data / 16.
        data -= data.mean(axis=0)

In [4]: # We learn the digits on the first half of the digits
        data_train, targets_train = (data[:n_samples // 2],
                                     digits.target[:n_samples // 2])


        # Now predict the value of the digit on the second half:
        data_test, targets_test = (data[n_samples // 2:],
                                   digits.target[n_samples // 2:])
        # data_test = scaler.transform(data_test)

In [5]: # Create a classifier: a support vector classifier
        kernel_svm = svm.SVC(gamma=.2)
        linear_svm = svm.LinearSVC()

In [6]: # create pipeline from kernel approximation
        # and linear svm
```

```
feature_map_fourier = RBFSampler(gamma=.2, random_state=1)
feature_map_nystroem = Nystroem(gamma=.2, random_state=1)
```

```
In [7]: fourier_approx_svm = pipeline.Pipeline([("feature_map", feature_map_fourier),
                                                ("svm", svm.LinearSVC())])

        nystroem_approx_svm = pipeline.Pipeline([("feature_map", feature_map_nystroem),
                                                ("svm", svm.LinearSVC())])
```

Código propio: hago el pipeline con un DecisionTree Nota: de momento uso el mismo RBF-Sampler que usan ellos, la misma instancia. Si afecta puedo usar uno nuevo. No lo creo

```
In [8]: fourier_approx_dt = pipeline.Pipeline([("feature_map", feature_map_fourier),
                                               ("dt", DecisionTreeClassifier())])
```

```
In [9]: # fit and predict using linear and kernel svm:

        kernel_svm_time = time()
        kernel_svm.fit(data_train, targets_train)
        kernel_svm_score = kernel_svm.score(data_test, targets_test)
        kernel_svm_time = time() - kernel_svm_time

        linear_svm_time = time()
        linear_svm.fit(data_train, targets_train)
        linear_svm_score = linear_svm.score(data_test, targets_test)
        linear_svm_time = time() - linear_svm_time
```

```
In [10]: sample_sizes = 30 * np.arange(1, 10)
         fourier_scores = []
         nystroem_scores = []
         fourier_times = []
         nystroem_times = []
```

Código propio: hago las listas para mi nuevo modelo. Nota: ellos usan un nombre ambiguo. Les llaman fourier_scores y fourier_times porque solamente la svm usa fourier. Nosotros usamos fourier con dt.

```
In [11]: fourier_dt_scores = []
         fourier_dt_times = []
```

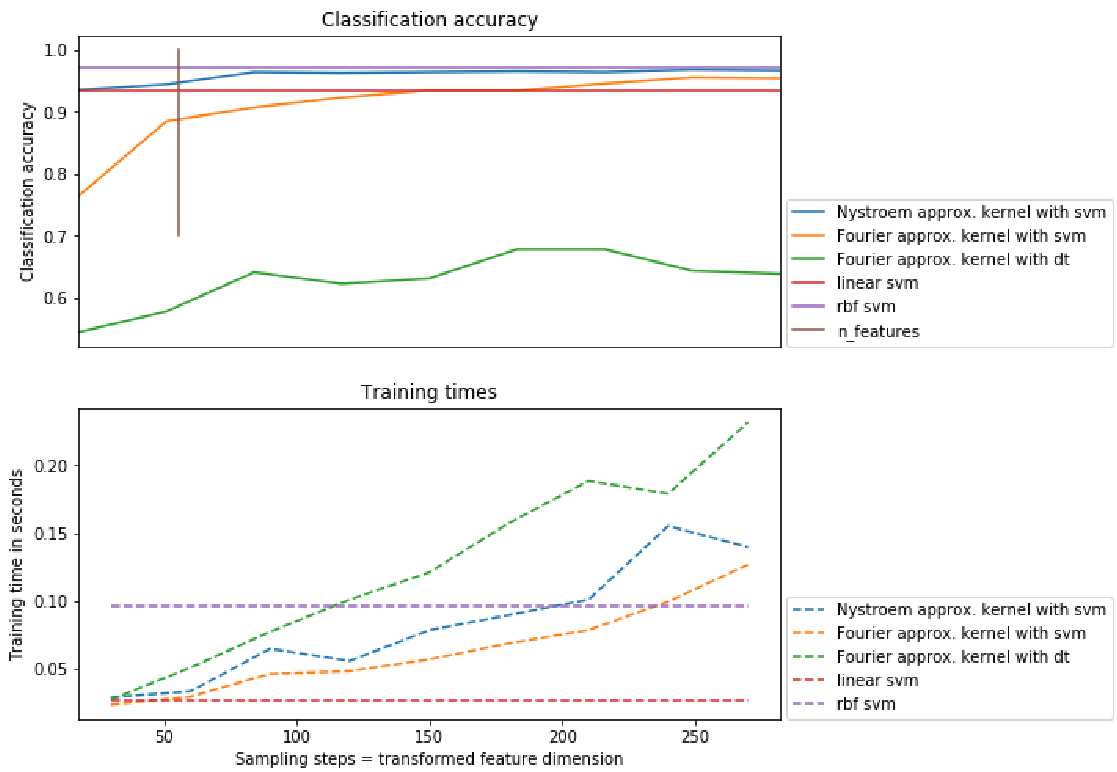
```
In [12]: for D in sample_sizes:
             fourier_approx_svm.set_params(feature_map__n_components=D)
             nystroem_approx_svm.set_params(feature_map__n_components=D)

             ##Código propio: lo mismo con mi pipeline
             fourier_approx_dt.set_params(feature_map__n_components=D)

             #Nystroem svm
             start = time()
```

```
        nystroem_approx_svm.fit(data_train, targets_train)
        nystroem_times.append(time() - start)

        # Fourier svm
        start = time()
        fourier_approx_svm.fit(data_train, targets_train)
        fourier_times.append(time() - start)

        # Código propio: el tiempo del fourier dt
        start = time()
        fourier_approx_dt.fit(data_train, targets_train)
        fourier_dt_times.append(time() - start)

        # Cálculo de scores
        fourier_score = fourier_approx_svm.score(data_test, targets_test)
        nystroem_score = nystroem_approx_svm.score(data_test, targets_test)

        # Código propio: calculo sl score con mi pipeline
        fourier_dt_score = fourier_approx_dt.score(data_test, targets_test)


        # Poner los scores en las listas de score
        nystroem_scores.append(nystroem_score)
        fourier_scores.append(fourier_score)

        # Código propio: pongo mis propios scores y times
        fourier_dt_scores.append(fourier_dt_score)
```

In [13]:
```
# plot the results:
plt.figure(figsize=(8, 8))
accuracy = plt.subplot(211)
# second y axis for timeings
timescale = plt.subplot(212)

# Accuracy y tiempo de nystroem
accuracy.plot(sample_sizes, nystroem_scores,
              label="Nystroem approx. kernel with svm")
timescale.plot(sample_sizes, nystroem_times, '--',
               label='Nystroem approx. kernel with svm')

# Accuracy y tiempo de fourier (este es ambiguo, se refiere al svm)
accuracy.plot(sample_sizes, fourier_scores,
              label="Fourier approx. kernel with svm")
timescale.plot(sample_sizes, fourier_times, '--',
               label='Fourier approx. kernel with svm')

# Accuracy y tiempo de fourier dt
accuracy.plot(sample_sizes, fourier_dt_scores,
```

```
                 label="Fourier approx. kernel with dt")
timescale.plot(sample_sizes, fourier_dt_times, '--',
                  label='Fourier approx. kernel with dt')

# horizontal lines for exact rbf and linear kernels:
accuracy.plot([sample_sizes[0], sample_sizes[-1]],
                 [linear_svm_score, linear_svm_score], label="linear svm")
timescale.plot([sample_sizes[0], sample_sizes[-1]],
                  [linear_svm_time, linear_svm_time], '--', label='linear svm')


accuracy.plot([sample_sizes[0], sample_sizes[-1]],
                 [kernel_svm_score, kernel_svm_score], label="rbf svm")
timescale.plot([sample_sizes[0], sample_sizes[-1]],
                  [kernel_svm_time, kernel_svm_time], '--', label='rbf svm')

# vertical line for dataset dimensionality = 64
accuracy.plot([64, 64], [0.7, 1], label="n_features")

# legends and labels
accuracy.set_title("Classification accuracy")
timescale.set_title("Training times")
accuracy.set_xlim(sample_sizes[0], sample_sizes[-1])
accuracy.set_xticks(())
#accuracy.set_ylim(np.min(fourier_scores), 1)
timescale.set_xlabel("Sampling steps = transformed feature dimension")
accuracy.set_ylabel("Classification accuracy")
timescale.set_ylabel("Training time in seconds")
accuracy.legend(loc=(1.01,0)) # Modifico la posición de la leyenda, que no se ve bien
timescale.legend(loc=(1.01,0))
```

```
Out[13]: <matplotlib.legend.Legend at 0x7ff30aec33c8>
```



In [14]:
```
## visualize the decision surface, projected down to the first
## two principal components of the dataset
#pca = PCA(n_components=8).fit(data_train)
#
#X = pca.transform(data_train)
#
## Generate grid along first two principal components
#multiples = np.arange(-2, 2, 0.1)
## steps along first component
#first = multiples[:, np.newaxis] * pca.components_[0, :]
## steps along second component
#second = multiples[:, np.newaxis] * pca.components_[1, :]
## combine
#grid = first[np.newaxis, :, :] + second[:, np.newaxis, :]
#flat_grid = grid.reshape(-1, data.shape[1])
#
## title for the plots
#titles = ['SVC with rbf kernel',
#           'SVC (linear kernel)\n with Fourier rbf feature map\n'
#           'n_components=100',
#           'SVC (linear kernel)\n with Nystroem rbf feature map\n'
#           'n_components=100']
#
```

```
#plt.tight_layout()
#plt.figure(figsize=(12, 5))
#
## predict and plot
#for i, clf in enumerate((kernel_svm, nystroem_approx_svm,
#                          fourier_approx_svm)):
#    # Plot the decision boundary. For that, we will assign a color to each
#    # point in the mesh [x_min, x_max]x[y_min, y_max].
#    plt.subplot(1, 3, i + 1)
#    Z = clf.predict(flat_grid)
#
#    # Put the result into a color plot
#    Z = Z.reshape(grid.shape[:-1])
#    plt.contourf(multiples, multiples, Z, cmap=plt.cm.Paired)
#    plt.axis('off')
#
#    # Plot also the training points
#    plt.scatter(X[:, 0], X[:, 1], c=targets_train, cmap=plt.cm.Paired,
#                edgecolors=(0, 0, 0))
#
#    plt.title(titles[i])
#plt.tight_layout()
#plt.show()
```