Graphical and Audio Features

Graphical Features

In addition to text and natural language processing, bag of words has successfully been applied to images by categorizing a collection of regions and describing only their appearance, ignoring any spatial structure. However this is not the typical approach used to represent images as features, and requires you come up with methods of categorizing image regions. More often used methods include:

1. Split the image into a grid of smaller areas, and attempt feature extraction at each locality. Return a combined array of all discovered. features
2. Use variable-length gradients and other transformations as the features, such as regions of high / low luminosity, histogram counts for horizontal and vertical black pixels, stroke and edge detection, etc.
3. Resize the picture to a fixed size, convert it to grayscale, then encode every pixel as an element in a uni-dimensional feature array.

Let's explore how you might go about using the third method with code:

```
# Uses the Image module (PIL)
from scipy import misc

# Load the image up
img = misc.imread('image.png')

# Is the image too big? Resample it down by an order of magnitude
img = img[::2, ::2]

# Scale colors from (0-255) to (0-1), then reshape to 1D array per pixel, e.g. grayscale
# If you had color images and wanted to preserve all color channels, use .reshape(-1,3)
X = (img / 255.0).reshape(-1)

# To-Do: Machine Learning with X!
#
```

If you're wondering what the :: is doing, that is called extended slicing. Notice the .reshape(-1) line. This tells Pandas to take your 2D image and flatten it into a 1D array. This is an all purpose method you can use to change the shape of your dataframes, so long as you maintain the number of elements. For example reshaping a [6, 6] to [36, 1] or [3, 12], etc. Another method called .ravel() will do the same thing as .reshape(-1), that is unravel a multi-dimensional

NDArray into a one dimensional one. The reason why its important to reshape your 2D array images into one dimensional ones is because each image will represent a single sample, and SKLearn expects your dataframe to be shapes [num_samples, num_features].

Since each image is treated as a single sample, your datasets likely need to have *many* image samples within them. You can create a dataset of images by simply adding them to a regular Python list and then converting the whole thing in one shot:

```
# Uses the Image module (PIL)
from scipy import misc

# Load the image up
dset = []
for fname in files:
  img = misc.imread(fname)
  dset.append(  (img[::2, ::2] / 255.0).reshape(-1)  )

dset = pd.DataFrame( dset )
```

Audio Features

Audio can be encoded with similar methods as graphical features, with the caveat that your 'audio-image' is already a one-dimensional waveform data type instead of a two-dimensional array of pixels. Rather than looking for graphical gradients, you would look for auditory ones, such as the length of sounds, power and noise ratios, and histogram counts after applying filters.

```
import scipy.io.wavfile as wavfile

sample_rate, audio_data = wavfile.read('sound.wav')
print audio_data

# To-Do: Machine Learning with audio_data!
#
```


English