Exercise 7

Due Friday July 02 2021.

Some files are provided that you need below: E7.zip. As usual, **you may not write any loops** in your code.

Dog Rates Significance

One last statistics question: when we looked at “Pup Inflation” in Exercise 2, we drew a fit line across the data, but didn't really ask the question at hand: have the ratings given by this Twitter account been changing over time?

Revisit your `dog-rates.ipynb` from exercise 2 and append some more useful results. **Output the p-value** from the regression for the question “is the slope different from zero?”. Also **plot a histogram** of the residuals (observed values minus predicted values). [Note the question about this below.]

CPU Temperature Regression

Let's revisit another question from the past. When we looked at CPU temperature values in Exercise 3, we made a prediction for the next temperature: could we have done better?

There was more data in my original data set. I actually collected CPU usage percent (at the moment the sample was taken), the system load *(https://en.wikipedia.org/wiki/Load_(computing))* (one minute average), fan speed (RPM), CPU frequency, and CPU temperature. Also, data was collected every 10 seconds: the Exercise 3 data set was subsampled down to once per minute. Surely we could have made a better “CPU temperature in the next time step” prediction with more data.

For this question, I have provided the data set as separate training and validation sets: `sysinfo-train.csv` and `sysinfo-valid.csv`. They should both be provided on the command line:

```
python3 regress_cpu.py sysinfo-train.csv sysinfo-valid.csv
```

Create a program `regress_cpu.py` based on the provided `regress_cpu_hint.py`. Things you need to do:

- Fill in the `'next_temp'` column in the DataFrame when it's read. This should be the temperature at time t+1, which is what we want to predict. Hint *(http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.shift.html)* .
- Create a scikit-learn linear regression model and fit it to the training data. Hint *(http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html)* , also do **not** fit an intercept (since there's nowhere to put it in the Kalman filter, as we have used it).
- Update the `transition` matrix for the Kalman filter to actually use the new-and-improved predictions for temperature.

Have a look at the output and note the questions below. When you submit, the **output should be** the one line from the `output_regression` function provided in the hint.

Colour Words

With previous classes, I have collected data mapping colours (specifically RGB colours *(https://en.wikipedia.org/wiki/RGB_color_model)* you saw on-screen) to colour words. When creating the experiment, I gave options for the English basic colour terms *(https://en.wikipedia.org/wiki/Color_term)* .

The result has been a nice data set: almost 4000 data points that we can try to learn with. It is included this week as `colour-data.csv`.

Let's actually use it for its intended purpose: training a classifier.

Create a program `colour_bayes.py`. You should take the name of the CSV file on the command line: the provided `colour_bayes_hint.py` does this and contains some code to nicely visualize the output.

Start by getting the data: read the CSV with Pandas. Extract the `X` values (the R, G, B columns) into a NumPy array and normalize them to the 0–1 range (by dividing by 255: the tools we use will be looking for RGB values 0–1). Also extract the colour words as `y` values.

Partition your data into training and validation sets using train_test_split *(http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html)* .

Now we're ready to actually do something: create a naïve Bayes classifier *(http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html#sklearn.naive_bayes.GaussianNB)* and train it. Use the default priors for the model: they are set from the frequency in the input, which seems as sensible as anything else.

Have a look at the accuracy score on the validation data to see how you did. Print the accuracy score for this model.

The score doesn't tell much of a story: call `plot_predictions` from the hint to see a plot of colours (left) and predicted colour categories (right).

Colour Words and Colour Distances

The naïve Bayes approach implicitly assumes that distances in the input space make sense: distances between training `X` and new colour values are assumed to be comparable. That wasn't a great assumption: distances between colours *(https://en.wikipedia.org/wiki/Color_difference)* in RGB colour space aren't especially useful.

Possibly our inputs are wrong: the LAB colour space *(https://en.wikipedia.org/wiki/Lab_color_space)* is much more perceptually uniform. Let's convert the RGB colours we have been working with to LAB colours, and train on that. The skimage.color *(http://scikit-image.org/docs/dev/api/skimage.color.html)* module has a function for the conversion we need. (You may have to install scikit-image, depending on your original setup).

We can create a pipeline *(http://scikit-learn.org/stable/modules/generated/sklearn.pipeline.make_pipeline.html#sklearn.pipeline.make_pipeline)* model where the first step is a transformer that converts from RGB to LAB, and the second is a Gaussian classifier, exactly as before.

There is no built-in transformer that does the colour space conversion, but with a function skimage.color that converts your `X` to LAB colours, you can create a FunctionTransformer *(http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.FunctionTransformer.html#sklearn.preprocessing.FunctionTransformer)* to do the work.

Have a look at the accuracy value for this model as well. When finished, **your `colour_bayes.py` should print two lines**: the first and second accuracy score. Please do **not** have a `plt.show()` in your code when you submit: it makes marking a pain.

Questions

Answer these questions in a file `answers.txt`.

1. Looking at your `dog-rates.ipynb`, do you think the residual are close-enough to being normal to look at the OLS p-value? Can you reasonably conclude that the ratings are increasing?
2. Do you think that the new “better” prediction is letting the Kalman filter do a better job capturing the true signal in the noise?

Submitting

Submit your files through CourSys for Exercise 7.

Updated Mon June 28 2021, 19:46 by ggbaker.