Ex6 Tutorials

Chirag UttamsinghMentorGeneral Discussion · 2 years ago · Edited by moderator

Here are the ex6 tutorials by Tom Mosher:

gaussianKernel():

The method is similar to that used for sigmoid.m in ex2. The numerator is the sum of the squares of the difference of two vectors. That's just like computing the linear regression cost function. Use exp() and scale by the value given in the formula (top of page 6 of ex6.pdf)

dataset3Params():

One method is to use two nested for-loops - each one iterating over the range of C or sigma values given in the ex6.pdf file.

Inside the inner loop:

- Train the model using svmTrain with X, y, a value for C, and the gaussian kernel using a value for sigma. See ex6.m at line 108 for an example of the correct syntax to use in calling svmTrain() and the gaussian kernel. Note: Use temporary variables for C and sigma when you call svmTrain(). Only use 'C' and 'sigma' for the final values you are going to return from the function.
- Compute the predictions for the validation set using svmPredict() with model and Xval.
- Compute the error between your predictions and yval.
- When you find a new minimum error, save the C and sigma values that were used. Or, for each error computation, you can save the C, sigma, and error values as rows of a matrix. When all 64 computations are completed, use min() to find the row with the minimum error, then use that row index to retrieve the C and sigma values that produced the minimum error.

==========

processEmail():

Start by putting a 'keyboard' command (or a breakpoint) in the processEmail.m script template, in the blank area below the "YOUR CODE HERE" comment block. Then run the ex6_spam script.

When program execution hits the "keyboard" command, it will break to the debugger, where you can inspect the variables "str" and "vocabList".

Observe that str holds a single word, and that vocabList is a cell array of all known words. Resume execution with the 'return' command.

For the code you need to add:

Here is an example using the strcmp() function showing how to find the index of a word in a cell array of words:

```
1
2
3
```

```
small_list = {'alpha', 'bravo', 'charlie', 'delta', 'echo'}
match = strcmp('bravo', small_list)
find(match)
```

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
strcmp() returns a logical vector, with a 1 marking each location where a word occurs in the list. If there is no match, all of the logical values are 0's.
The find() function returns a list of the non-zero elements in "match". You can add these index values to the word_indices list by concatenation.
Note that if there is no match, find() returns a null vector, and this can be concatenated to the word list without any problems.
Note that your word_index list must be returned as a column vector. If you make it a row vector, the submit grader will still give you credit, but your emailFeatures() function will not work correctly.
==========
emailFeatures:
The emailFeatures() function is one of the simplest in the entire course:

- You're given a list of word indexes.
- For each index in that list, you're asked to set the corresponding entries in an 'x' array to the value '1'.

A couple of different methods could be used:

- Loop through the list of word indexes, and use each index to set the corresponding value in the 'x' array to 1.
- Take advantage of vectorized indexing, and do the same operation in one line of code without the loop.

Note that the 'x' feature list must be a column vector, and the word_indices list (which is provided by your processEmail() function) must be a column vector.
You can complete this function by adding only one line of code.Try this example in your console:

```
1
2
3

vec = zeros(10,1)     % included in the function template
indexes = [1 3 5]     % you're provided with a list of indexes
vec(indexes) = 1      % set the values to 1
```

END