Typesetting

January 23, 2018

1 Markdown

This is a markdown cell which can be used to type text.

LaTeX can be written inline by surrouding the LaTeX with single $: $x \cdot y$

Surrounding with double $$ allows you to write LaTeX on its own line:

$$x \cdot y$$

See the documentation for more information and commands.

2 Sample Code and Output

```
In [1]: def dotProduct(d1, d2):
            """
            @param dict d1: a feature vector represented by a mapping from a feature (string)
            @param dict d2: same as d1
            @return float: the dot product between d1 and d2
            """
            if len(d1) < len(d2):
                return dotProduct(d2, d1)
            else:
                return sum(d1.get(f, 0) * v for f, v in d2.items())

In [2]: def increment(d1, scale, d2):
            """
            Implements d1 += scale * d2 for sparse vectors.
            @param dict d1: the feature vector which is mutated.
            @param float scale
            @param dict d2: a feature vector.
            NOTE: This function does not return anything, but rather
            increments d1 in place. We do this because it is much faster to
            change elements of d1 in place than to build a new dictionary and
            return it.
            """
            for f, v in d2.items():
                d1[f] = d1.get(f, 0) + v * scale
```

```
In [3]: x = {'the':1.0, 'a':0.5}
        y = {'the':1.0, 'a':0.5, 'or': .3}

In [4]: dotProduct(x,y)

Out[4]: 1.25
```

3 Converting to pdf

Follow the instructions here to install the nbconvert package and all dependencies (pandoc and TeX).

The generated pdf will be from the last save point so make sure you save all your changes before running the following cell.

```
In [5]: #Run this cell in the the notebook to generate the pdf
        ! jupyter nbconvert --to pdf Typesetting.ipynb

[NbConvertApp] Converting notebook Typesetting.ipynb to pdf
[NbConvertApp] Writing 22071 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 30599 bytes to Typesetting.pdf
```

Alternatively, you can use the command line and run this command from the directory where the notebook is saved: `jupyter nbconvert -to pdf Typesetting.ipynb`

More information can be found here.

3.1 Line Continuations

When using nbconvert, long lines of code may be truncated. To avoid this, use line contiunations to make sure a single line of code is not too long. This is good practice for code readiabilty as well.

Python has implied line continuations inside parenthesis, brackets, and braces. You can also use the character for explicit line continuations.

See the examples below.

```
In [6]: import pandas as pd
        import numpy as np

In [7]: #implicit line continuation
        df = pd.DataFrame(np.random.randint(low=0, high=10, size=(5, 5)),
                          columns=['a', 'b', 'c', 'd', 'e'])

In [8]: #explicit line continuation
        a = 1   \
            + 2 \
            + 3 \
            - 4
```