Preparing Students for Data Science: Filling in the Gaps

Dennis Sun
Cal Poly, San Luis Obispo and Google
dsun09@calpoly.edu

Joint Statistical Meetings
July 29, 2018

CAL POLY

CAL POLY

My Background

- I am a statistics professor at Cal Poly, where I primarily teach data science courses.
- I also work (part-time) as a data scientist at Google.
- Naturally, I am interested in the question of how to prepare students for a career in data science.

Course Background

- My ideas in this talk are inspired by DATA 301, an "Intro to Data Science" course that I have been developing at Cal Poly.
- DATA 301 is a first course for students interested in a career in data science, <u>not</u> a data literacy course.
- Class is taken primarily by Computer Science and Statistics majors. Prerequisites are: 2 CS courses and 1 STAT course.

Question:

What should a “Data Science” course teach that is not already covered by existing computer science and statistics courses?

CAL POLY

What is Data Science?

“Data science is the intersection of statistics and computer science.”

To some extent, this is true:

- Data scientists need to know about sampling, randomized experiments, and statistical inference.
- Data scientists also need to know how to implement algorithms, query data from databases, and distribute computations over a cluster of machines.

But there are also topics that are not covered by existing statistics and computer science courses. I will focus on those gaps today.

CAL POLY

The Structure of Tabular Data

DataFrame of the 2010 Boston Marathon results

In [3]:
```
marathon.head()
```

Out[3]:

	division	name	city	gender	age	official	bib	overall	state	genderdiv	net	country
0	1 / 22	Van Dyk, Ernst	Paarl	M	37	86.88	W1	1 / 29	NaN	1 / 24	NaN	RSA
1	3 / 4660	Merga, Deriba	Addis Ababa	M	29	128.65	1	3 / 22672	NaN	3 / 13120	NaN	ETH
2	3 / 4996	Kosgei, Salina	Eldoret	F	33	148.58	F1	58 / 22672	NaN	3 / 9552	NaN	KEN
3	2 / 22	Schabort, Krige	Cedartown	M	46	86.93	W2	2 / 29	GA	2 / 24	NaN	USA
4	4 / 22	Masazumi, Soejima	Fukuoka	M	39	88.10	W3	4 / 29	NaN	4 / 24	NaN	JPN

other

quantitative (discrete)

quantitative (continuous)

categorical

- **Rows** represent observations. Thinking about the observational unit is important.
- **Columns** represent variables. Knowing the variable types is important.

Cal Poly

The DataFrame: A Special Data Structure

In CS classes, students learn about data structures, like arrays and hash maps.

The DataFrame is a specialized data structure that is highly customized for data analysis:

- It is not just an array of arrays because columns (variables) need to be just as easily accessible as rows (observations).
- It is not just a matrix because different columns can contain different types (and within a column, types need to be consistent).
- In computer science, the primitive data types are integers, floats, strings, etc., but in data science, the data types are quantitative, categorical, etc.

These are **programming ideas** that CS students are unlikely to have encountered in their other courses!

CAL POLY

Data as First-Class Citizen

In data science, functions and data structures for working with data are first-class citizens.

To determine the female marathon winner, many students write code like this:

```
fastest_time = 100000
for i, row in marathon.iterrows():
  if row["gender"] == "F" and row["official"] < fastest_time:
    fastest_time = time
```

when this should be a one-liner:

```
marathon[marathon["gender"] == "F"]["official"].min()
```

The convoluted solution is the result of thinking like a general programmer. Data science programming is different.

Operations like subsetting and minimizing are first-class citizens in data science, just like loops and conditionals are first-class citizens in general programming.

CAL POLY

Translating Calculations into Code

- I gave students a data set of OKCupid profiles (published by Albert Kim in the Journal of Statistics Education).
- I asked them to calculate and interpret the conditional distributions of sexual orientation given sex.

```
In [3]: okcupid = pd.read_csv("data/okcupid/profiles.csv")
        okcupid
```

essay2	essay3	...	location	offspring	orientation	pets	religion	sex	sign	smokes	speaks	status
making people laugh. \nranting about a go...	the way i look. i am a six foot half asian, ha...	...	south san francisco, california	doesn’t have kids, but might want them	straight	likes dogs and likes cats	agnosticism and very serious about it	m	gemini	sometimes	english	single
being silly. having ridiculous amonts of fun w...	NaN	...	oakland, california	doesn’t have kids, but might want them	straight	likes dogs and likes cats	agnosticism but not too serious about it	m	cancer	no	english (fluently), spanish (poorly), french (...	single
improvising in different contexts. alternating...	my large jaw and large glasses are the physica...	...	san francisco, california	NaN	straight	has cats	NaN	m	pisces but it doesn’t matter	no	english, french, c++	available
playing synthesizers and	socially		berkeley	doesn’t		likes					english,	

Distribution of Orientation Given Sex

Easiest way to do this is to first calculate a contingency table:

In [4]:
```
counts = pd.crosstab(okcupid["orientation"], okcupid["sex"])
counts
```

Out[4]:

sex	f	m
orientation		
bisexual	1996	771
gay	1588	3985
straight	20533	31073

and normalize by the totals for each sex:

In [5]:
```
counts.sum()
```

Out[5]:
```
sex
f    24117
m    35829
dtype: int64
```

Distribution of Orientation Given Sex

In all, just two lines of code:

In [6]:
```
counts = pd.crosstab(okcupid["orientation"], okcupid["sex"])
counts / counts.sum()
```

Out[6]:

sex	f	m
orientation		
bisexual	0.082763	0.021519
gay	0.065846	0.111223
straight	0.851391	0.867258

Of course, there are more manual ways to do this (e.g., `for` loop over the cells of the table), but this is probably the most elegant, generalizable, and efficient way.

Statistics vs. Data Science

In a statistics course, we might give students the contingency table:

	Sex	
	f	m
bisexual	1996	771
gay	1588	3985
straight	20533	31073

and ask them to calculate the conditional distribution of orientation given sex (by hand).

In a data science course, we expect students to handle the additional complexity of translating the calculation into (efficient) code.

Computational Thinking

What does `counts / counts.sum()` do?

`counts`

	f	m
	1996	771
	1588	3985
	20533	31073

`counts.sum()`

f	24117
m	35829

The division aligns the vector with the columns of the matrix...

`counts`

f	m
1996	771
1588	3985
20533	31073

/

`counts.sum()`

f	m
24117	35829
24117	35829
24117	35829

... and **broadcasts** the vector across the rows.

Students have to master two things:

- figuring out what calculation to do
- implementing that calculation in code

CAL POLY

Data Representations

Most statistical methods and software packages have been designed with tabular data in mind.

But modern data increasingly comes in non-tabular formats:

- **hierarchical**: XML and JSON
- **textual**: raw text
- **spatial**: shapefiles
- etc.

Students need to be able to work with these different kinds of data. In particular, they may want to convert this data to tabular form to take advantage of existing tools.

Example: Textual Data

How would we convert a corpus of text documents to tabular data?

1. `I am Sam. Sam I am....` $\Longrightarrow [0, 2, 1, 0, 5, ...]$
2. `One fish. Two fish.`
   `Red fish. Blue fish....` $\Longrightarrow [1, 4, 1, 0, 0, ...]$
3. `At the far end of town where`
   `the Grickle-grass grows...` $\Longrightarrow [0, 2, 0, 8, 0, ...]$
4. `The sun did not shine.`
   `It was too wet to play....` $\Longrightarrow [0, 1, 4, 0, 0, ...]$

The numbers in the table represent word frequencies (possibly normalized or reweighted). For example, one common scheme is **TF-IDF**.

A good way to measure similarities between two vectors of word frequencies is **cosine similarity**.

Where would a student learn this?

- Statistics class? Probably not.
- Machine learning class? Maybe, if the class dealt with textual data.
- They would see this in a Natural Language Processing or an Information Retrieval course. But can we expect everyone to take such a specialist course?

A concept as fundamental as what to do with different data representations belongs in a generalist course!

Conclusion

I have argued that there are gaps in our existing computer science and statistics curricula that are necessary for data science:

1. data science programming (with distinct data structures and first-class citizens from general programming)
2. translating calculations into code
3. working with diverse data representations (e.g., textual, hierarchical, spatial)

A budding data scientist will still have to delve deeply into computer science and statistics, but they also need data science courses that fill in these gaps.

I am working on an "Principles of Data Science" textbook based on these ideas (to be released in 2019), so my thinking continues to evolve, and I welcome your feedback!

Thank You!

Dennis Sun (`dsun09@calpoly.edu`)

CAL POLY