probability_plots

March 12, 2021

1 Probability Plots

In this lecture, we'll be using another case study to help us learn about probability plots. In particular, we'll be focusing on something called a Quantile-Quantile plot (QQ plot), which is a graphical technique that allows us to see whether a particular data set follows a given distribution. We'll also discuss how to use a statistical test to make quantitative determinations as to whether or not a sample is indeed drawn from a particular distribution.

Now, one of the benefits of learning data science from a large public research university like the University of Michigan is that we're able not just to share with you different techniques and approaches, but to ground those techniques in our own work. As you might know, my research is in an area broadly known as *learning analytics*, which is the study of learning, teaching, and education through data and analytics processes. So in this lecture I'm going to share some of our own work and how we applied visual exploration and statistical methods to understand and evaluate educational predictive models.

1.1 Motivation: Educational Predictive Models

One area of current are of interest in education involves developing predictive models for use in early warning systems. So, for instance, we may want to predict how a student might perform in a class and identify those who may be at risk for dropping out of a course. The hope is to use this information in order to stage an appropriate intervention and provide that student with the necessary support and resources to succeed. At the University of Michigan we actually have one of these systems called *Student Explorer*, which was created by a colleague of mine, Stephanie Teasley, who is faculty here in the School of Information. You can see from the diagram above that it uses a number of different features about the student grade, the course average, and the site page views to classify the student into one of three categories, and it uses a sort of red yellow green signal light metaphor when communicating this to advisors depending upon the risk category the student is in.

Many of these early warning systems use private and sensitive student data such as your grades, demographics, online activity logs, and other details. As a result, there are have concerns about how this data is being used and the potential implications, including:

- personal privacy concerns
- legislative oversight
- and risks and inequities

While universities have many protocols, procedures, and structure to keep your data safe and secure, numerous news stories on data breaches certainly do not instill confidence in people regarding how their data is being managed. For education specifically, there are also pieces of legislation, such as FERPA, which restricts how personal identifying information is to be used and shared.

And so this brings you to the research issue I've been studying, with my PhD student Warren Li and collaborator Florian Schuab who is faculty here in the School of Information. Specifically, if students explicitly opt-out of having their data shared for these purposes, how would that affect the quality of predictive models we were able to build? Would it add bias to the models across groups like differnt ethnic groups or students in different years in their programs? Would the models degrade to the point where they are no longer usable and, how fast would this degradation happen?

So, we started to explore this, and collected data on how students would opt out, and how it might affect the quality of predictive models

1.2 Analyzing Grade Predictions from a Model

[ ]:
```
# I can't share the original data, and so we won't implement the machine␣
 ↪learning model here. Instead I've
# have provided a select subgroup of grades and predictions from our model␣
 ↪which I'll use to demonstrate our
# analysis approach. Let's load that into a DataFrame.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
original_df = pd.read_pickle("assets/mads_data_pre.pkl")

# We're also going to define two constant lists: one for the valid letter␣
 ↪grades, and another for their
# corresponding grade points. These will be used later on.
VALID_GRADES = ['A', 'A-', 'B+', 'B', 'B-', 'C+', 'C', 'C-', 'D+', 'D', 'D-',␣
 ↪'E']
NUMERIC_GRADES = [4.0, 3.7, 3.3, 3.0, 2.7, 2.3, 2.0, 1.7, 1.3, 1.0, 0.7, 0.0]

# Now let's look at the DataFrame
original_df.head()
```

[ ]:
```
# Now, we need a way to organize this information and get a visual sense of␣
 ↪what our model is doing. To do so,
# let's plot the actual grades student's received against the predicted grades.␣
 ↪So for example, how many times
# did our model predict an A grade when the student actually received an A,␣
 ↪versus a B, and so forth?

# To do this, I'm going to build something called a confusion matrix
def plot_confusion_matrix(cm, classes,
```

```
                          normalize=False):

    cm = np.float_(cm)
    # if the normalize flag is set then we'll just change the data values␣
 ↪through scaling
    if normalize:
        cm = cm / cm.sum(axis=1)[:, np.newaxis]
    # now, the plot I'm going to show is actually just called imshow, and it's␣
 ↪used to show images, or pixel
    # data. this is basically what will be passed in by cm, since it's a square␣
 ↪matrix. So we're essentially
    # rendering a bunch of pixels to the screen, a grid of values
    plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
    plt.colorbar()
    # notice that I've set the cmap. This is the color map we want to use to␣
 ↪show values, and you can read
    # about it in the docs. My PhD student Warren particularly likes blues, so␣
 ↪this is why we're using these
    # values

    # Now let's set a title and some axes grades
    plt.title('Confusion matrix')
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

    # And lets update the X and Y tick_marks. We expect these to be the grades␣
 ↪which are passed in
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
```

[ ]:
```
# Ok, we have the mechanis of plotting a square matrix of values down now. This␣
 ↪will come in handy, not just
# here but any time you are interested in looking at a confusion matrix, which␣
 ↪is essentially a square
# matrix of your actual values versus predicted values. This helps you identify␣
 ↪where error might lie in your
# model

# Speaking of, let's create that matrix. We have a dataframe of results, but we␣
 ↪want to now aggregate that
# into a list of true grades versus predicted grades. The library sklearn,␣
 ↪which you'll use in the machine
# learning course, has a nice function to create this kind of matrix
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(original_df['Actual'], original_df['Predicted'])
cm
```

[ ]:
```
# Ok, that's interesting! We see big numbers in some places and small numbers␣
 ↪in other places. It's not
# super meaningful to us yet, so let's use our plotting function
plot_confusion_matrix(cm, VALID_GRADES)
```

[ ]:
```
# Notice that regardless of the number of true labels, we almost always predict␣
 ↪that a student will receive
# either an A- or an A! What's with that? Let's take a look at the frequencies␣
 ↪of our data
original_df.groupby('Actual').apply(len)
```

[ ]:
```
# Ok, so we see that in this dataset there are way more "A grade" predictions␣
 ↪(over 100,000!) when compared to
# any other grade. Why is that?
```

1.3 Visualizing Distributions: Quantile-Quantile Plots

To try and figure out why this is happening, let's try running some **diagnostics**, and try to get a better understanding regarding our underlying data. Specifically, we're going to make a QQ plot, which stands for Quantile-Qunatile plot.

Now, from the boxplots and violinplots discussion you should be familiar with Quartiles, which are points that separate out 25% (or one quarter) of the data. Quantiles (also known as percentiles) are just a generalization of this idea. So the 0.5 quantile would be the 50th percentile, and so half of the data would lie below/above this point.

The formula to create a QQ plot is: 1. We order our n data points in ascending order (we're making each point it's own quantile) 2. We evenly divide a normal distribution into n+1 segments (each with equal area) 3. We compute the z-values for each of these "cut-off" points (theoretical quantiles) 4. We plot the actual quantiles from step 1 against the theoretical quantiles in step 3

Now, the open textbook I've shared has a discussion of this under the term Quantile-Normal plot, since they are specifically interested in checking whether a given set of data is normally distributed and and they call these QN plots. In the most general case, however, you can use any distribution of data for the theoretical quantiles. You can find this on page 83, and here's a link to that textbook again: https://www.stat.cmu.edu/~hseltman/309/Book/chapter4.pdf

And just reflecting for a moment on these four steps, it means that you're creating a scatter plot where one dimension is the normal distribution, and the other is your actual distribution. This means a straight line of points means your data follows the same distribution.

If you want to see this in more detail, here are a couple of additional videos which describe the statistics. But for the rest of this demonstration I want to show you how to write the code to do this in matplotlib. * https://www.youtube.com/watch?v=IFKQLDmRK0Y * https://www.youtube.com/watch?v=okjYjClSjOg

[ ]:
```
# So, you won't need to manually go through the process I just described,␣
 ↪instead we can just use the
# proability plot library from scipy.stats

# let's import stats
import scipy.stats as stats
```

```
# Now to get plots like this we want to convert our letter grades into numeric␣
 ↪equivilents, so let's just
# create a mapping dictionary
grade_point_dict = dict(zip(VALID_GRADES, NUMERIC_GRADES))
# And now let's apply that dictionary to our dataframe, replacing values as␣
 ↪appropriate
grade_dist = original_df.replace({'Actual' : grade_point_dict})['Actual']

# We're going to create two plots here on two different axes. Don't worry about␣
 ↪this, think of each just as
# their own figure
plt.figure(figsize=(12,5))
ax1 = plt.subplot(121)
ax2 = plt.subplot(122)

# Ok, now we can pass this list of grades into stats.probplot(). This function␣
 ↪takes the distribution we
# want to compare against, and we'll use stats.norm for a normal distribution,␣
 ↪and a location for the plot,
# and we'll indicate that we want it on the first axis, ax1
stats.probplot(grade_dist, dist=stats.norm, plot=ax1)

# Now let's plot next to it a histogram of the grades. Pandas plotting also can␣
 ↪take an axis to just drop a
# plot, and this is why understanding matplotlib is so important in a python␣
 ↪world -- most libraries which
# offer plotting support out of the box do so based on matplotlib
grade_dist.hist(ax=ax2)
```

[ ]:
```
# Ok, so time for some interpretation. First, in the histogram on the right we␣
 ↪see that this is very much not
# a normal distribution. All of values skew to the right, very few people end␣
 ↪up getting a 0, for instance.
# on the left we see our QQ plot. The red line indicates the theoretical␣
 ↪quantiles, and the blue dots show our
# actual values. These clearly do not line up.
```

[ ]:
```
# I wonder if this is an exponential distribution instead?
# Let's just take a look by running another probplot()
stats.probplot(grade_dist, dist=stats.expon, plot=plt.gca())
```

[ ]:
```
# Nope, doesn't look like it.

# To demonstrate what happens if we do have something that looks more␣
 ↪bell-curve shaped, here's an example
# of some fake data
```

```
curved = ([4.0]*1 + [3.7]*2 + [3.3]*2 + [3.0]*3 + [2.7]*5 + [2.3]*7 + [2.0]*7 +␣
 ↪[1.7]*5 + [1.3]*3 +
          [1.0]*2 + [0.7]*2 + [0.0]*1)
stats.probplot(curved, dist=stats.norm, plot=plt.gca())
```

[ ]:
```
# Notice how the points we have fall nicely along that theoretical quantile␣
 ↪line.
```

Ok, so back to the problem we actually faced. Now, we could leave the lecture there but I want to dig deeper and show you our next steps. The issue was that there was significantly more data for learners with high grades, and so our model is unable to predict other types of grades reliably. In other words, if we just predicted that everyone would get an 'A', we'd have a pretty good accuracy already! Even though that wouldn't make for a very useful predictive model. To combat this, we can try to balance out our dataset and try undersampling. You'll learn more about different sampling techniques in a later course, but this essentially just means that we reduced the size of our dataset so that each class we were looking to predict – the letter grade, had basically the same number of students in it. Actually, it's not quite this simple, we used a techniques called SMOTE – synthetic monitority oversampling technique – to generate a bunch of dake data for model training, but let's leave that for the future and stick with undersampling.

[ ]:
```
# So, let's read in our new predictions after building this new model
processed_df = pd.read_pickle("assets/mads_data_post.pkl")

# And now let's turn that into a confusion matrix
cm=confusion_matrix(processed_df['Actual'], processed_df['Predicted'])

# And let's plot that
plot_confusion_matrix(cm, VALID_GRADES)
```

[ ]:
```
# Ok, now we see we get a much more broad range of values! And now we can plot␣
 ↪a QQ plot of our predicted
# grades and take a look at how it stacks up to the normal distribution

# Let's makes sure we are looking at the predicted grades distribution this␣
 ↪time
grade_dist = processed_df.replace({'Predicted' : grade_point_dict})['Predicted']

# And we'll copy and paste out plotting code
plt.figure(figsize=(12,5))
ax1 = plt.subplot(121)
ax2 = plt.subplot(122)

# And let's put up a probability plot
stats.probplot(grade_dist, dist=stats.norm, plot=ax1)

# And the histogram
grade_dist.hist(ax=ax2)
```

[ ]:
```
# So we see that while not exactly normally distributed, this set of grades␣
 ↪does cover a larger area. And if
# you're interested more in what we did in this work to quantify the quality of␣
 ↪the model you can check out
# the paper linked in the course as an optional reading.
```

1.4 Statistical Tests: Comparing Distributions

So, in this class we're focused on the visual exploration of data, and I've showed you a few ways we've explored data in this paticular research work, where the first was to build a simple heatmap of pixels representing a confusion matrix, then to build quantile-quantile plots and historgrams to compare our distribution of data against theoretical distributions, in this case the normal distribution.

But, becuase this is a real analysis, I want to go beyond this visual analysis into the follow up statistical analysis we did. And I want to do this in part because it relates a lot to our visual analysis.

If we have the null hypothesis H_0, that our distribution is normal, and our alternative hypothesis H_a, that our distribution is not normal, we can conduct a Kolmogorov-Smirnov (or KS) test to determine whether or not we should reject the null hypothesis. Similar to how we can find a t-statistic when we do t-tests, we can also calculate a KS test statistic.

I'm going to paste in here the formal definition is given for reference, but you won't be expected to understand the details, and the material in this subbox is purely optional. Wikipedia has a great article on this approach:

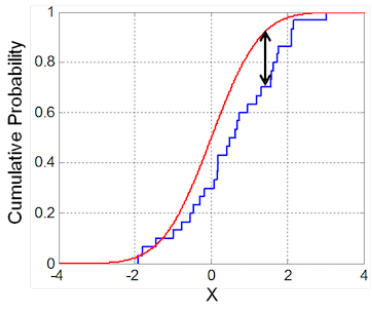
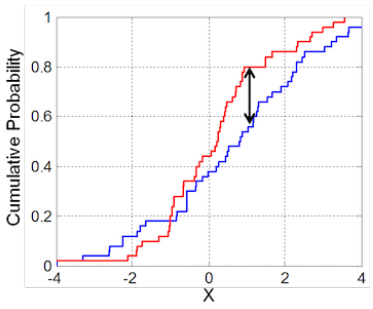
$$D_n = \max_x |F_n(x) - F(x)|$$

where $F(x)$ is a cumulative distribution function (cdf) and $F_n(x)$ is an empirical distribution function given by

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} I_{[-\infty,x]}(X_i)$$

and $I_{[-\infty,x]}$ is an indicator function equal to 1 if $X_i \leq x$ and 0 otherwise.

I think that the main idea can be illustrated by this diagram from wikipedia,

one-sample KS test	two-sample KS test
	

We can see if the cumulative distribution function (CDF) is compareable to a distribution we're interested in, such as a normal distribution. This is just like our QQ plot! However, in this case, the test statistic is calculated by considering the largest distance between these two sets, as shown by the arrows in the figure. And so the KS test measures this distance. And you can do this investigation either versus theoretical distribution data (the one-sample KS) or between two different samples (the two-sample KS test).

[ ]:
```
# Let's generate some synthetic experiment data
experimental_data=sorted(np.random.chisquare(6,size=1000))
# And let's say we have a baseline we want to comare this data to
theoretical_data=sorted(np.random.normal(size=1000))
# And then we can look at the two plots overlayed
plt.plot(experimental_data)
plt.plot(theoretical_data)
```

[ ]:
```
# Now let's assume the experimental data is a population of students we gave␣
 ↪some magic treamtent to, and the
# theoretical data is actually something we observed in another classroom. Are␣
 ↪the distribution of scores
# between these students different? Well, we don't have to write the KS test␣
 ↪ourselves, it exists within the
# scipy.stats library
print(stats.ks_2samp(experimental_data, theoretical_data))
```

[ ]:
```
# The result of the KS test is two values, the KS statistic and the pvalue. You␣
 ↪can see here that the
# statistic is pretty close to one and the pvalue is very small, so it looks␣
 ↪like these are indeed from two
# different distributions
```

[ ]:
```
# Recall that we talked about some of the risks of bias in predictive models␣
 ↪earlier on. Here's an example of
# the actual predictive models we trained, and this shows the performance of␣
 ↪our predictive models when
# predicting on male and female students. Each point represents a different␣
 ↪amount of simulated opt-out,
# evenly-spaced from 0-99%. So this is real data from our study
```

```
gender_df = pd.read_pickle('assets/mads_data_gender.pkl')
gender_df.head()
```

[ ]:
```
# As an aside, the actual metric we're using for quality of a model is called␣
 ↪Cohen's Kappa, and you'll learn
# about that in the future. A 1 means that the predictive model is good, a 0␣
 ↪means we're basically guessing.
# These scores really aren't that good, frankly.
```

[ ]:
```
#We want to see whether there is actually a difference between these columns of␣
 ↪data, so we plotted it
gender = ['Male', 'Female']
drop_percentages = np.linspace(0.00, 0.99, 25)

plt.figure(figsize=(12,6))
scores_by_gender = {gender: [x for x in gender_df[gender]] for gender in␣
 ↪gender_df.columns}
for label, y_arr in scores_by_gender.items():
    plt.plot(
        drop_percentages,
        y_arr,
        label=label
    )
plt.ylim(ymin=0, ymax=1)
plt.title('Opt-out Effect on Kappa for Decision Trees by Gender')
plt.ylabel('$\kappa$')
plt.xlabel('Proportion of data dropped from training set')
lgd = plt.legend(loc='upper left')
```

[ ]:
```
# Looking at the plot, it seems like there is not much difference between our␣
 ↪ability to predict grades for
# male students versus female students. In this case that's a sign of a model␣
 ↪that has minimal gender bias.
# But we can also use the statistical tools we just learned to answer this␣
 ↪question more definitively.
print(stats.ks_2samp(scores_by_gender['male'], scores_by_gender['female']))
```

[ ]:
```
# And there you see a pretty high p value, suggesting that we can't reject the␣
 ↪null hypothesis
```

1.5 Conclusions

Ok, this was a big lecture. We've talked about probability plots and statistical tests to help us compare different distributions. While we specifically mentioned QQ plots and KS tests, there are also other graphical tools such as P-P plots and tests such as the Shapiro-Wilk or Anderson-Darling test, each with their own advantages and disadvantages. However, I've tried to equip you with all the key ideas behind these techniques and we encourage you to explore additional ones

if you'd like.

In the full paper of this study, we found that there's actually quite a significant range of opt-out that schools have to consider when developing these systems, which vary depending on how you count people who don't respond to your data sharing requests, and even whether the question wording is simply changed from "opt-in" to "opt-out". And we actually do in fact spot some bias: we are better at predicting grades for students who are in their fourth year of their studies versus freshman. But but this gap quickly narrows as opt-out increases. If you're interested in learning more, I've provided a link to the paper here: https://dl-acm-org.proxy.lib.umich.edu/citation.cfm?id=3303809