2017-01-30

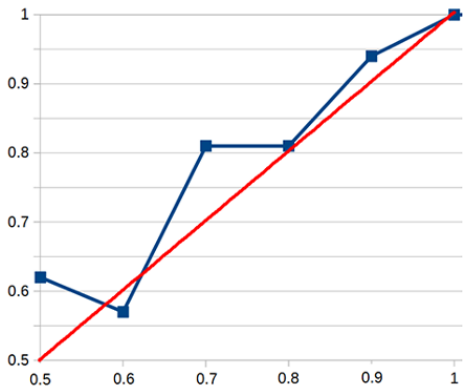
Prediction Calibration - Doing It Right

第一

To improve your skill at predicting the future, you work on two things - accuracy and calibration. There are well established and mathematically pretty ways to score accuracy. However, scoring calibration tends to be a mixture of eyeballing histograms and calculating fractions of failed predictions for arbitrary fixed ranges of confidence. This is not ideal.

First, let's talk about the current ("traditional") method, and what is wrong with it.

As an example, take a look at the Slate Star Codex prediction calibration post from the end of 2016. The relevant fragment is:

> *Of 50% predictions, I got 8 right and 5 wrong, for a score of 62%*
>
> *Of 60% predictions, I got 12 right and 9 wrong, for a score of 57%*
>
> *Of 70% predictions, I got 13 right and 3 wrong, for a score of 81%*
>
> *Of 80% predictions, I got 13 right and 3 wrong, for a score of 81%*
>
> *Of 90% predictions, I got 16 right and 1 wrong, for a score of 94%*
>
> *For 95% predictions, I got 9 right and 0 wrong, for a score of 100%*
>
> *For 99% predictions, I got 3 right and 0 wrong, for a score of 100%*
>
> *This is the graph of my accuracy for this year:*
>
> 
>
>
> *Red is hypothetical perfect calibration, blue is my calibration. I am too lazy and bad at graphs to put in 95% right, but it doesn't change the picture very much (especially because it's impossible to get a very accurate 95% with 9 questions).*
>
> *The 50% number is pretty meaningless, as many people have noted, so my main deviation was some underconfidence at 70%. This was probably meaningless in the context of this year's numbers alone, but looking back at 2014 and 2015, I see a pretty similar picture. I am probably generally a little bit underconfident in medium probabilities (I have also gotten lazier about making graphs).*

Blog Archive

January 2018 (3)
December 2017 (3)
November 2017 (1)
October 2017 (2)
September 2017 (1)
August 2017 (2)
July 2017 (1)
May 2017 (2)
April 2017 (1)
March 2017 (2)
January 2017 (2)
November 2016 (1)
October 2016 (1)
September 2016 (2)
August 2016 (1)
April 2016 (1)
March 2016 (1)

More by SquirrelInHell

- AI Safety Comics
- Android Apps
- Be Well Tuned
- Rationality Updates

第二

What we see in the SSC post quoted above seems to be a picture-perfect incarnation of the "traditional" method. Arbitrary confidence levels or ranges, check. A graph or histogram based strictly on those arbitrary levels, check.

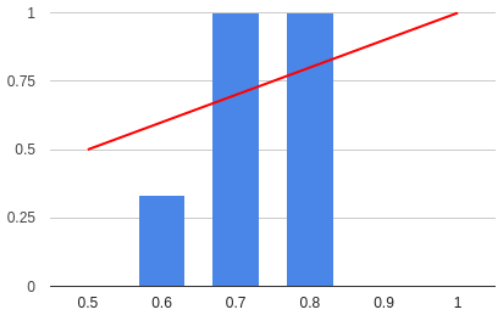
The first thing that is wrong with it, is that even before making the predictions, Scott had to restrict himself to only claiming certain levels of confidence that he knew would be easy to calculate later (80%, 90%, 95%, 99% etc.).

The second thing that is wrong with it, is that he needs quite a few predictions *in each category* to make the calculations be about anything other than pure noise. Note: of course, to get a meaningful result we need a lot of data. But we don't need to require a lot of data *in each separate arbitrarily chosen subcategory*.

The third thing that is wrong with it, is that it does not degrade gracefully. Imagine that you have made only a few predictions so far, and you are wondering how you are doing. Those few data points cannot give you any strong evidence about your calibration - but they can give you *some*. Imagine trying to make sense of these 6 results:

> *Of 60% predictions, I got 1 right and 2 wrong, for a score of 33%*
>
> *Of 70% predictions, I got 2 right and 0 wrong, for a score of 100%*
>
> *Of 80% predictions, I got 1 right and 0 wrong, for a score of 100%*

And the following bar chart:



Not very useful, isn't it. And it's not getting any better if you change it to a line chart.

The fourth thing that is wrong with it, is that the graph doesn't show the *relative* strenth of evidence about calibration in different ranges of confidence. That is, a category with fewer data points looks the same as a category with more, but it actually has more noise and is less important for your overall judgement.

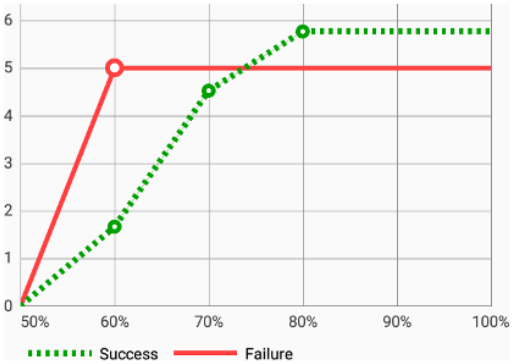
I could probably keep on complaining, but I'm sure you are getting the point by now.

第三

What is the solution? I am not claiming that my answer is the best possible one, but compared to previous state-of-the-art, it certainly does seem revelatory.

The idea is, in summary, to plot the most likely number of predictions that you would be expected to have made, if someone were to guess it from your actual numbers until a certain confidence level, doing this based on failed and successful predictions *separately*. The closer the two results would be to each other, the better your calibration.

This way of plotting turns out to have *quite* a few advantages over the "traditional" method. Before we go into that, let's look at an example, using the made-up data from above (only 6 data points).



You have 2 failed and 1 successful prediction at 60%, 2 successful at 70%, and 1 successful at 80%. So your successes are slightly more concentrated in the higher probabilities - but only *slightly*. The curves are still pretty close to each other, considering the small absolute values. You probably have nothing to worry about here.

More or less, the smaller the area between the two curves, the better your calibration. Note that the values on the graph are *comparable*, in a plain apples-to-apples sense.

Contrast with the useless bar chart above, made using the "traditional" method from the same data.

What are the advantages?

- You can use a different confidence level (73%, 99.1% etc.) for each prediction, and your results will not be worse for it (or harder to calculate).
- The graph will change slightly if you change your confidence level slightly (there is no "jump" between categories).
- The method is applicable for any number of predictions, and smootly gets better as you get more data points.
- The strength of evidence is represented on the graph (how quickly the absolute values grow).
- The shape of the graph also gives you an immediate sense of in which ranges you made how many predictions in total.
- The method makes sense mathematically, and seems to agree with intuition in all cases where intuition has something to say (this is a matter of personal taste, though).

第四

Well then, how does one make these improved graphs? Is it not too much trouble?

First note that this is already implemented in my prediction tracking app for Android, so you might just want to use it.

If you are still interested, here we go. It's actually quite simple:

- If you have some correct predictions at a confidence level p > 0.5, one would expect you to make on average 1 / p predictions to get each success.
- So the value of the success curve at a point 0.5 < x <= 1 is the sum of 1 / p for all p's of successful predictions such that 0.5 < p <= x.
- If you have some failed predictions at a confidence level p > 0.5, one would expect you to make on average 1 / (1-p) predictions to get each failure.
- So the value of the failure curve at a point 0.5 < x <= 1 is the sum of 1 / (1-p) for all p's of failed predictions such that 0.5 < p <= x.

Note that predictions made with 50% confidence can't be meaningfully labeled as "failed" or "successful". If you want to include them, make sure that they contribute an equal value to both curves.

I'm also adding linear interpolation between significant data points, which makes the graph somewhat easier to read and doesn't break things too much in cases that matter. But I guess a stepped line or a point plot would be a more accurate representation of the math.

9 comments:

Gunnar Zarncke Tuesday, January 31, 2017

Great approach. I installed the app (great that it is so compact) and starting to experiment with it. One thing is very disappointing: If I create a new entry and forget to set a date or time I get a warning and the prediction is lost :-(Also I got a message that the time was too early though it wasn't and I also lost the prediction.

Reply

Replies

クリス Tuesday, January 31, 2017

Thanks for reporting the issue. I've added a fix now, you'll probably get an update from Google Play within the next day or so.

Gunnar Zarncke Tuesday, January 31, 2017

Wow! That was fast!

Some feedback on how I (want to) use it (I'm doing this on paper currently):

- I make daily predictions of tasks I'm planing for the day. In most cases the prediction is for the next day and thus the due time is always the next morning. I guess many people might do the majority of their predictions with a fixed schedule. Maybe support a default day/time?

- Many predictions I do I copy from the previous day - namely if I didn't complete them. I make a new prediction whether I will make it this day (often with a different confidence). I guess many people will make comparable predictions for different days/weeks/months. Maybe an option to copy a prediction (with auto-forwarded due-date) would be nice.

SquirrelInHell Wednesday, February 01, 2017

As for the first request, acknowledged but I'm not promising anything.

As for the second, this is is already possible if you long tap a prediction and choose "Try Again".

Reply

Gustavo Thursday, February 16, 2017

App seems really nice! I will give it a roll. If I find your approach really clearer than the usual one, I will start recommending it instead of my own Predict app http://lesswrong.com/r/discussion/lw/mmp/predict_log_your_predictions_app/

Do you plan on opening the source code?

Reply

Replies

SquirrelInHell Thursday, February 16, 2017

Let me know how it goes, and if you have any suggestions on how to further improve the app.

@ open source: Not currently. If it happens, then I will still keep some of the additional/advanced features private, to have the option of creating a paid version of the app.

Reply

Rick Sunday, April 16, 2017

Could you post the actual algorithm, formally (I'm not sure how to reconstruct the graph you provided from your description of it)?

Reply

Replies

SquirrelInHell Sunday, April 16, 2017

OK, here's some pseudocode - I hope it is more clear now!

```
// input:

// PREDS - an array that for each confidence level,

// contains the number of predicted outcomes at that level

int[1..99] PREDS := ...

int SUM_CORRECT := 0

int SUM_FAILED := 0

for i = 51 to 99

begin

SUM_CORRECT += PREDS[i] * 100 / i

SUM_FAILED += PREDS[100-i] * 100 / (100-i)

drawDataPoint("CORRECT", i, SUM_CORRECT)

drawDataPoint("FAILED", i, SUM_FAILED)

end
```

Reply

Rene Roundthecorner Wednesday, October 25, 2017

Really nice. I will try out the app for sure (~99% confidence) :)

Reply

Subscribe to: Post Comments (Atom)