Error bars

It's good practice to give some indication of uncertainty or expected variability in experimental results. You will need to report experimental results in your coursework. Many of you will also write up experimental results in dissertations this year, and you will also want to know how seriously to take numbers that you measure in your future work.

We will discuss some different "standard deviations" that you might see reported, or want to report, including "standard errors on the mean".

1 Standard errors on a mean

Imagine we are taking a series of experimental measurements $\{x_n\}_{n=1}^N$. These could be all sorts of things: for example, the times taken for different runs of a program that you're testing, or the weights of some people who attend a particular gym.

We will assume that the measurements are taken independently from some unknown distribution. In the first example, the test conditions for the computer program are stable, and the times depend on independent random choices. In the second example, the people were selected at random from members of the gym. These distributions are 'unknown' in that we don't have mathematical descriptions of them. However, we can draw samples (gather data) from the distributions in these examples.

The mean μ and variance σ^2 of the distribution are usually unknown, but we assume they are finite. We can estimate them from the sample mean and variance, given N independent samples:

$$\mu \approx \bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_n \tag{1}$$

$$\sigma^2 \approx \hat{\sigma}^2 = \frac{1}{N-1}\sum_{n=1}^{N}(x_n - \bar{x})^2. \tag{2}$$

The $(N-1)$ in the estimator for the variance, rather than N, (Bessel's correction) is a small detail you don't need to worry about for this course.[1]

The estimator $\bar{x}$ is itself a random variable: if we gathered a second dataset and computed its mean in the same way, we would get a different $\bar{x}$. For some datasets $\bar{x}$ will be bigger than the underlying true mean μ, sometimes it will be smaller. The mean of $\bar{x}$ is the correct answer μ. That is, $\bar{x}$ is an unbiased estimator.

Using the rules of expectations and variances (see the note in the background section), we can estimate the variance of $\bar{x}$. We assume here that the observations are independent:

$$\begin{aligned}\mathrm{var}[\bar{x}] &= \frac{1}{N^2}\sum_{n=1}^{N}\mathrm{var}[x_n] &(3)\\ &= \frac{1}{N^2}N\sigma^2 = \sigma^2/N \approx \hat{\sigma}^2/N. &(4)\end{aligned}$$

A "typical" deviation from the mean is given by the standard deviation (*not* the variance). So we write:

$$\mu = \bar{x} \pm \hat{\sigma}/\sqrt{N}, \tag{5}$$

to give an indication of how precisely we think we have measured the mean of the distribution with our N samples. Some papers might report ± two standard deviations.

[1] The $(N-1)$ normalization makes the variance estimator unbiased and is what the Matlab/Octave `var` function does by default. NumPy's `np.var` requires the option `ddof=1` to get the unbiased estimator. However, if N is small enough that this difference matters, you need to be more careful about the statistics than we are in this note.

If the distribution over observations has finite mean and variance, then for large N the Central Limit Theorem (CLT) tells us that $\bar{x}$ will be approximately Gaussian distributed close to its mean. We *don't* assume that the data are Gaussian distributed. We just note that the sum, and so average, of many values will be approximately Gaussian. With that interpretation, we expect the estimate $\bar{x}$ to be within 1 standard error $\sigma/\sqrt{N}$ about 2/3 of the time, and within 2 standard errors about 95% of the time.

Care: we don't evaluate the 'true' standard error $\sigma/\sqrt{N}$, but an approximation of it, $\hat{\sigma}/\sqrt{N}$. Moreover, the CLT will only be accurate for large N, and can't be trusted several standard deviations away from the mean. In other words, the "error bar" $\mu = \bar{x} \pm \hat{\sigma}/\sqrt{N}$ gives some indication of what means are plausible. Giving this statement is better than just stating $\mu \approx \bar{x}$, but more concrete statistical statements would require additional analysis.

2 Application to test set errors

The average test set loss,

$$L_{\text{test}} = \frac{1}{M}\sum_{m=1}^{M} L(y^{(m)}, f(\mathbf{x}^{(m)})) = \frac{1}{M}\sum_{m=1}^{M} L_m, \qquad (6)$$

is an estimate of the generalization error, the average loss we would see if we could gather an infinite test set with the same distribution. How wrong might this estimate be?

We don't assume that the individual losses, $L(y, f(\mathbf{x}))$, are Gaussian-distributed, and they often aren't. For example, when performing classification we might report the 0–1 loss, which is zero when we are correct and one when we make an error. In this example, the distribution over the losses is a Bernoulli distribution.

However, we can compute the empirical mean and variance of any set of losses, and report an estimate of the mean, with a standard error bar.

Before taking such an error bar on test performance too seriously we would need to think about whether the theory above applies. Are the test cases independent? If the loss isn't bounded, is it likely to have a finite variance that we can reasonably estimate? Will future inputs to our model come from the same distribution? Will the relationship between inputs and outputs that we're modelling continue to be the same?

3 Reliability of a method

A standard error on the test set loss indicates how much the future performance of a particular *fitted model* might deviate from the performance we have estimated. It doesn't say whether the machine learning *method* would work well in future, if training were run again to create a new model.

Readers of a paper may also want to know how variable the performance of a model can be across different fits, that is, how robust is the *method*? The fitted models could vary for multiple reasons: we might gather new data; some machine learning methods depend on random choices; somewhat horrifyingly, even machine learning code using no random numbers and running on the same data, often gives different results![2] To summarize one of these effects, we could report the standard deviation of the models' performances (not a standard error on the mean) to indicate how much a future fit will typically vary from average performance when something is changed.

Important: Papers are sometimes not clear on what their "error bars" are reporting. Sometimes they show the standard deviation of results under different conditions, other times a standard error indicating uncertainty of the generalization error due to a finite test set. Always try to be clear precisely what standard deviation/error you are reporting and why.

[2] This phenomenon is mentioned in passing in `https://arxiv.org/abs/1707.05589`. Different single-point precision round-off errors occur when different choices are made by a parallel scheduler on a GPU, and the effect can be as large as changing a random initialization!

4 Which model is better?

If we fit two models, A and B, we can get a test set loss and standard error for both. If these two error bars overlap, it is common — but usually wrong — to conclude that we can't tell whether A is better than B.

As an example, if A had a loss of 0.1 less than B on every single test case in a test set of 1,000 cases, I would be incredibly confident that A was better than B. The two standard errors for the models' performance could be larger (for example 0.5). It is possible to be sure of the ordering of two models, without knowing how good either is very precisely.

To test model A against model B we could do a simple paired comparison. Construct the difference in losses on each test case:

$$\delta_m = L(y^{(m)}, f(\mathbf{x}^{(m)}; B)) - L(y^{(m)}, f(\mathbf{x}^{(m)}; A)). \tag{7}$$

If the mean of the δ's is several standard errors greater than zero, we would report that A is the better model. (Non-examinable: you could perform a paired t-test if you wanted to turn this idea into a formal hypothesis test.)

5 Test your understanding

I hope you'll try computing error bars "for real", when you next gather some data. However, you can (and should!) check you know how on a toy example now.

For example, try generating 100 variables from some non-Gaussian distribution with a mean you know. For example a Bernoulli distribution:

```
    xx = 1 * (np.random.rand(100) < 0.3)
```

Now estimate the mean from the 100 data-points, along with a standard error bar on that mean. While `scipy` has a standard error function, you should compute the error bar from the standard deviation at first, so you definitely know what you are computing and why. If you re-run your code several times, is the error bar usually reasonable?

Not everything you read will have an explicit suggestion of something to try at the end. You should come up with these ideas yourself: I often look for ways to check my mathematical understanding (which is often wrong). If there is a small special case where we know the answer, trying it out on a computer is often a good idea.

6 Further Reading and Reflection

This course doesn't get into any "proper" or sophisticated statistical analysis of results. For example, if you wanted to do formal statistical tests on approximately normally distributed quantities, a t-test takes into account that we can only estimate the variance of the distribution.

However, there is no point talking about t-tests, or other specific statistical tests, if it's not clear what you are doing and why!

This note reflects what is actually the most important first step: think about what sources of variability are affecting your results, and get some idea of how big these variations can be. Sometimes the difference in performance of two models is so much larger than any source of variability, that careful testing is not required. If careful testing is required, you should consider whether anyone cares about the size of the difference, even if it's "statistically significant". If the performance of models is close, maybe we should pick the model that's easiest to implement or maintain, or takes less computer time.