Statistical methods in NLP
Estimation

Richard Johansson

UNIVERSITY OF GOTHENBURG

January 28, 2016

why does the teacher care so much about the coin-tossing experiment?

- because it can model many situations:
  - I pick a word from a corpus: is it *armchair* or not?
  - is the email is spam or legit?
  - do the two annotators agree or not?
  - was the document correctly classified or not?

example: error rates

- a document classifier has an error probability of 0.08
- we apply it to 100 documents
- what is the probability of making exactly 10 errors?
- what is the probability of making 5–12 errors?

error rates: solution with Scipy

```
import scipy.stats

true_error_rate = 0.08
n_docs = 100

experiment = scipy.stats.binom(n_docs, true_error_rate)

print('Probability of 10 errors:')
print(experiment.pmf(10))

print('Probability of 5-12 errors:')
print(experiment.cdf(12) - experiment.cdf(4))
```

statistical inference: overview

- estimate some parameter:
  - what is the **estimate** of the error rate of my tagger?
- determine some interval that is very likely to contain the true value of the parameter:
  - 95% **confidence interval** for the error rate
- test some hypothesis about the parameter (not today):
  - is the error rate **significantly** greater than 0.03?
  - is tagger A significantly better than tagger B?

overview

estimating a parameter

interval estimates

random samples

- a **random sample** $x_1, \ldots, x_n$ is a list of values generated by some random variable X
  - for instance, X represents a die, and the sample is $[6, 2, 3, 5, 4, 3, 1, 3, 6, 1]$
- typically generated by carrying out some repeated experiment
- examples:
  - running a PoS tagger on some texts and counting errors
  - word and sentence lengths in a corpus

estimating a parameter from a sample

- given a sample, how do we estimate some parameter of the random variable that generated the data?
  - often the 'heads' probability p in the binomial
  - but the parameter could be anything: mean, standard deviation, ...
- an **estimator** is a function that looks at a sample and tries to guesses the value of the parameter
- we call this guess a **point estimate**: it's a single value
  - we'll look at intervals later

making estimators

- there are many ways to make estimators; here are some of the most important recipes
  - the **maximum likelihood** principle: select the parameter value that maximizes the probability of the data
  - the **maximum a posteriori** principle: select the most probable parameter value, given the data and my preconceptions about the parameter
  - **Bayesian** estimation: model the distribution of the parameter if we know the data, then find the mean of that distribution

maximum likelihood estimates

- select the parameter value that maximizes the probability of the sample $x_1, \ldots, x_n$
- mathematically, define a **likelihood function** $L(p)$ like this:

$$L(p) = P(x_1, \ldots, x_n | p) = P(x_1 | p) \cdot \ldots \cdot P(x_n | p)$$

- then find the p_{MLE} that maximizes $L(p)$
- this is a general recipe; now we'll look at a special case

ML estimate of the probability of an event

- we carry out an experiment n times, and we get a positive outcome x times
  - for instance: we classify 50 documents with 7 errors
- how do we estimate the probability p of a positive outcome?

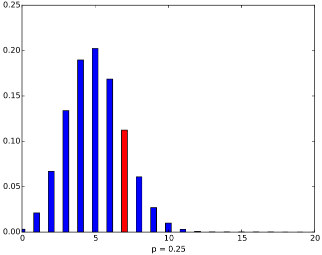
estimating the probability

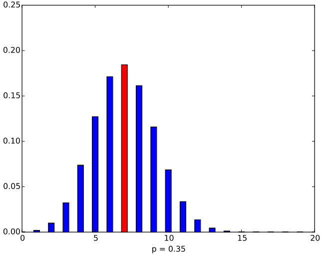
- this experiment is a binomial r.v. with parameters n and p
- ML estimation: find the p_{MLE} that **makes x most likely**
- that is, we find the p that maximizes

$$L(p) = P(x|p) = \binom{n}{x} \cdot p^x \cdot (1-p)^{n-x}$$

maximizing the likelihood

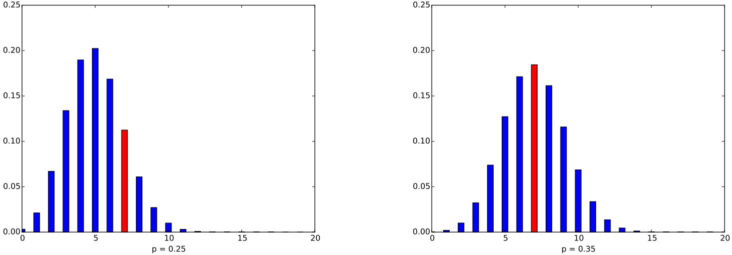
- we classified 20 documents with 7 errors; what's the MLE of the error rate?





maximizing the likelihood

- we classified 20 documents with 7 errors; what's the MLE of the error rate?



- it can be shown that the value of p that gives us the maximum of $L(p)$ is

$$p_{MLE} = \frac{x}{n}$$

UNIVERSITY OF GOTHENBURG

example: ML estimate of the probability of a word

- we observer the words in a corpus of 1,173,766 words
- I see the word *dog* 10 times
- assuming a unigram model of language: what the probability of a randomly selected word being *dog*?

example: ML estimate of the probability of a word

- we observer the words in a corpus of 1,173,766 words
- I see the word *dog* 10 times
- assuming a unigram model of language: what the probability of a randomly selected word being *dog*?
- ML estimate:

$$p_{MLE}(\text{dog}) = \frac{10}{1173766}$$

evaluating performance of NLP systems

- when evaluating NLP systems, many performance measures can be interpreted as probabilities
  - error rate and accuracy:
    - accuracy $= P(\text{correct})$
    - error rate $= P(\text{error})$
  - precision and recall:
    - precision $= P(\text{true} \mid \text{guess})$
    - recall $= P(\text{guess} \mid \text{true})$
- we estimate all of them using MLE

information retrieval example

- there are 1,752 documents about cheese in my collection
- I type *cheese* into the Lucene search engine and it returns a number of documents, out of which 1,432 are about cheese
- what's the estimate of the recall of this information retrieval system?

information retrieval example

- there are 1,752 documents about cheese in my collection
- I type *cheese* into the Lucene search engine and it returns a number of documents, out of which 1,432 are about cheese
- what's the estimate of the recall of this information retrieval system?

$$R_{MLE} = \frac{1432}{1752} = 0.817$$

maximum a posteriori estimates

- **maximum a posteriori** (MAP) is a recipe that's an alternative to MLE
- the difference: it takes our **prior beliefs** into account
  - "coins tend to be quite even, so you need to get 'heads' *many* times if you're going to persuade me!"

maximum a posteriori estimates

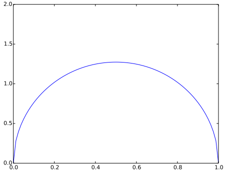
- **maximum a posteriori** (MAP) is a recipe that's an alternative to MLE
- the difference: it takes our **prior beliefs** into account
  - "coins tend to be quite even, so you need to get 'heads' *many* times if you're going to persuade me!"
- instead of just the likelihood, we maximize the **posterior probability** of the parameter:

$$\underset{\text{posterior}}{\arg\max_p \underbrace{P(p|\text{data})}} = \arg\max_p \underset{\text{prior}}{P(p)} \cdot \underset{\text{likelihood}}{P(\text{data}|p)}$$

- we use the prior to encode what we believe about p
  - if I make no assumption (uniform prior), MLE = MAP

MAP with a Dirichlet prior

- the **Dirichlet** distribution is often used as a prior in MAP estimates
- assume that we pick n words randomly from a corpus
  - the words come from a vocabulary with the size V
  - we saw the word *armchair* x times out of n



- with a Dirichlet prior with a concentration parameter α, the MAP estimate is

$$p_{MAP} = \frac{x + (\alpha - 1)}{n + V \cdot (\alpha - 1)}$$

- for instance, with $\alpha = 2$, we get

$$p_{MAP} = \frac{x + 1}{n + V}$$

UNIVERSITY OF GOTHENBURG

overview

estimating a parameter

interval estimates

interval estimates

- if we get some estimate by ML, can we say something about how reliable that estimate is?
- a **confidence interval** for the parameter p with significance value α is an interval $[p_{low}, p_{high}]$ so that

$$P(p_{low} \leq p \leq p_{high}) \geq \alpha$$

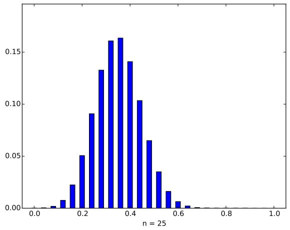
- for instance: with 95% probability, the error rate of the spam filter is in the interval $[0.05, 0.08]$
  - that is: it is between 0.05 and 0.08

interval estimates: overview

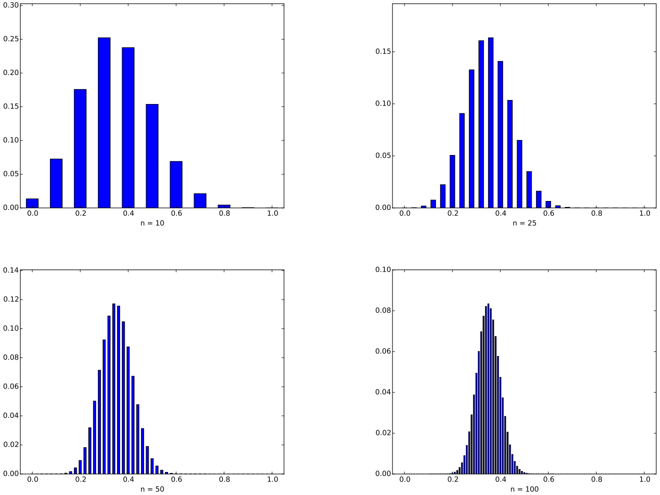
- in this course, we will mainly use confidence intervals when we evaluate the performance of some system
- we will now see a cookbook method for computing confidence intervals for probability estimates, such as
  - classifier error rate
  - precision / recall for a classifier or retrieval system
- for more complex evaluation metrics, we'll show a more advanced technique in later lectures
  - part-of-speech tagging accuracy
  - bracket precision / recall for a phrase structure parser
  - word error rate in ASR
  - BLEU score in machine tranlation

the distribution of our estimator

- our ML or MAP estimator applied to randomly selected samples is a random variable with a distribution
- this distribution depends on the **sample size**
  - large sample $\rightarrow$ more concentrated distribution
- we will reason about this distribution to show how a confidence interval can be found

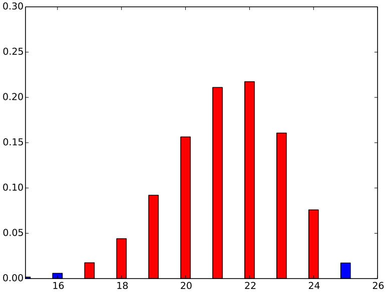


estimator distribution and sample size ($p = 0.35$)



the distribution of ML estimates of heads probabilities

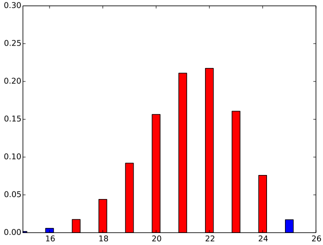
- if the true p value is 0.85 and we toss the coin 25 times, what results do we get?



- 95% of the experiments give a result between 17 and 24
- ...so 95% of the estimates will be between 0.68 and 0.96

confidence interval for the ML estimation of a probability

- assume we toss the coin n times, with x 'heads'
- cookbook recipe for computing an approximate 95% confidence interval $[p_{low}, p_{high}]$:
  - first estimate $p^* = x/n$ as usual
  - to compute the lower bound p_{low}:
    1. let X be a binomially distributed r.v. with parameters n, p^*
    2. find x_{low} = the 2.5% percentile of X
    3. $p_{low} = x_{low}/n$
  - for the upper bound p_{high}, use the 97.5% percentile instead



in Scipy

- assume we got 'heads' x times out of n
- recall that we use ppf to get the percentiles!

```
p_est = x / n

rv = scipy.stats.binom(n, p_est)

p_low = rv.ppf(0.025) / n
p_high = rv.ppf(0.975) / n

print(p_low, p_high)
```

example: political polling

- I ask 38 randomly selected Gothenburgers about whether they support the congestion tax in Gothenburg
- 22 of them say *yes*
- an approximate 95% confidence interval for the popularity of the tax is 0.421 – 0.737

```
number_yes = 22
total_number = 38
p_est = number_yes / total_number

rv = scipy.stats.binom(total_number, p_est)
p_low = rv.ppf(0.025) / total_number
p_high = rv.ppf(0.975) / total_number

print(p_low, p_high)
```

don't forget your common sense

- I ask 14 MLT students about whether they support the congestion tax, 11 of them say *yes*
- will I get a good estimate?
- in NLP: the "WSJ fallacy"

next time

- exercise in the computer lab
- you will estimate some probabilities and compute confidence intervals