Statistical methods in NLP
Evaluating and comparing classifiers

UNIVERSITY OF GOTHENBURG

Richard Johansson

February 16, 2016

overview

overview of the second programming assignment

significance testing for the accuracy

comparing two classifiers

the second programming assignment

- in the previous assignment, you implemented a classifier
- you will now make a more thorough analysis of how well it works

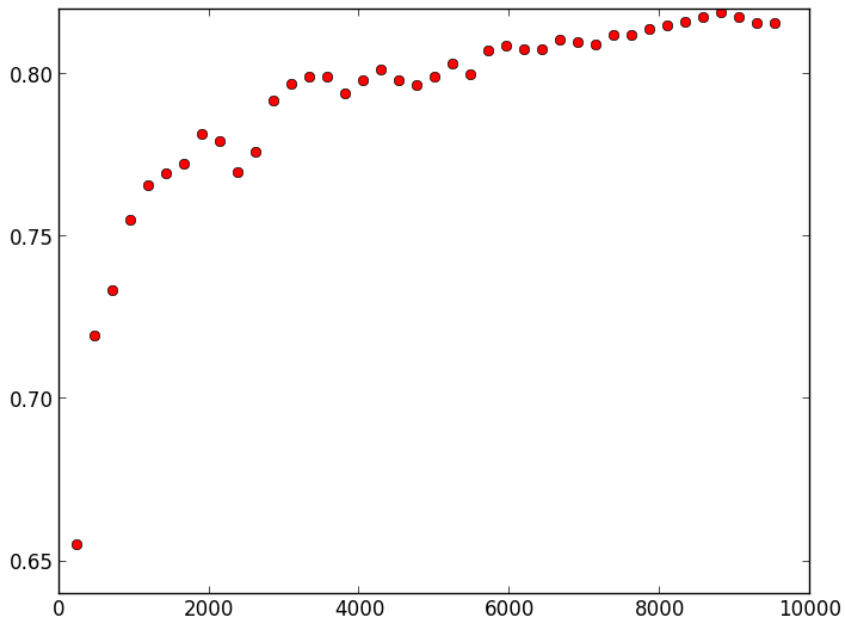
estimating the accuracy, and precision/recall

- warmup: compute the accuracy (as in last assignment)
- also: precision and recall for the positive class

$$P = \frac{\text{nbr. correct pos}}{\text{nbr. guessed pos}} \qquad R = \frac{\text{nbr. correct pos}}{\text{nbr. pos in the test set}}$$

learning curve

- set aside a test set
- use training sets of different sizes
- estimate the accuracy for each training set size
- plot the result



confidence interval for the accuracy

- you can use the recipe that we used previously:

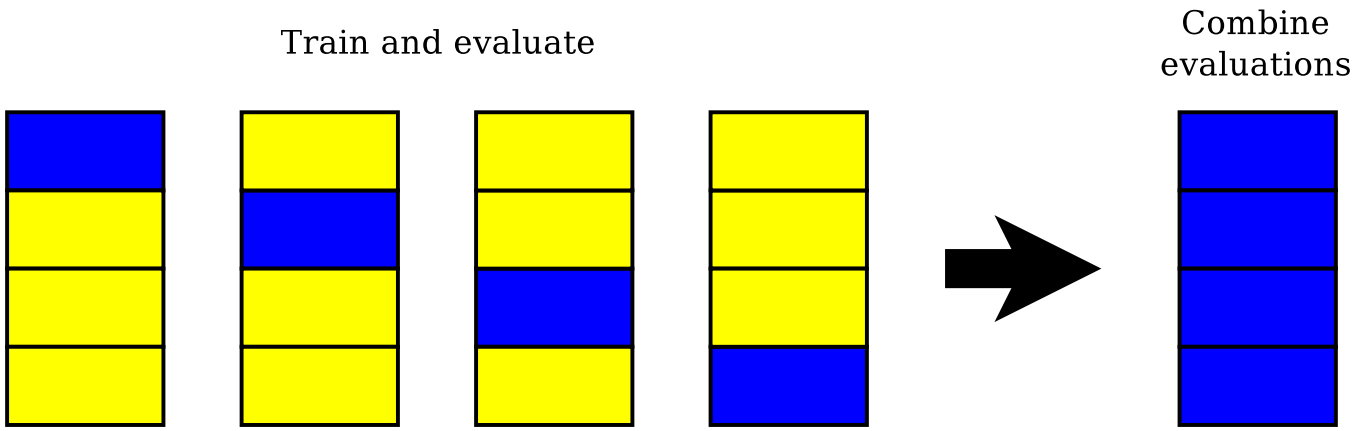
```
acc_mle = n_correct / n

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

acc_low = rv.ppf(0.025) / n
acc_high = rv.ppf(0.975) / n
```

evaluation with cross-validation

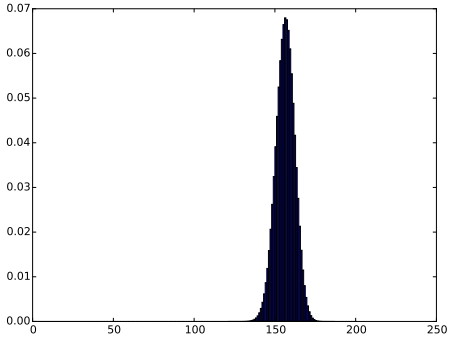
- we've discussed that the uncertainty of the estimate depends on the test set size
- **cross-validation** allows us to use more data for testing
- idea:
  - split into N parts of equal size (**folds**)
  - each fold is once a test set; then the others are the training set
  - evaluate on each fold and combine evaluations

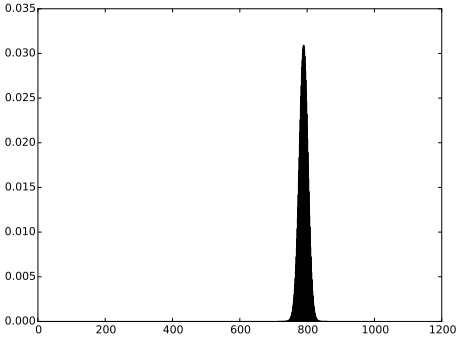


- recompute the accuracy and its confidence interval

cross-validation example

- our corpus consists of 1,000 documents, we split it into five 200-document pieces
- we use part 1 for testing and train on parts 2, 3, 4, 5; we classify 156 instances correctly
- for the other parts, the number of correct guesses is 163, 149, 164, and 157, respectively
- if we only consider part 1, then the CI is 0.720–0.835
- if we consider all parts, then the CI is 0.763–0.814





statistical significance testing for the accuracy

- in the last two tasks, you will consider two questions:
  - how sure are we that the true accuracy is better than 0.80?
  - how sure are we that classifier A is better than classifier B?
- we'll see recipes that can be used in these two scenarios
- these recipes work when we can assume that the "tests" (e.g. documents) are independent
  - fine for evaluations of classifiers, but not for more complex linguistic evaluations (tagging, parsing, translation, etc)

overview

overview of the second programming assignment

significance testing for the accuracy

comparing two classifiers

comparing the accuracy to some given value

- my boss has told me to build a classifier with an accuracy of at least 0.70
- my NB classifier made 40 correct predictions out of 50
  - so the MLE of the accuracy is 0.80
- based on this experiment, how certain can I be that the accuracy is really greater than 0.70?
- if the true accuracy is 0.70, **how unusual is our outcome**?

statistical hypothesis testing

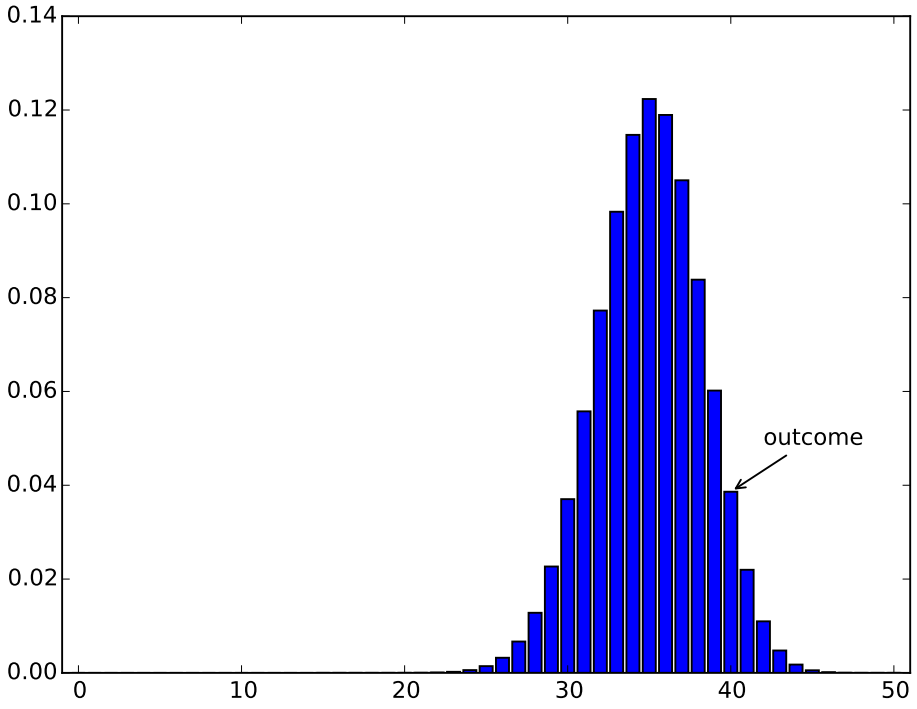
- we assume a **null hypothesis** and then see how **unusual** our outcome is
  - the null hypothesis is typically "boring": the true accuracy is equal to 0.7
- the "unusualness" is measured by the ***p*-value**
  - if the null hypothesis is true, how likely are we to see an outcome as unusual as the one we got?
- we typically want our p-values to be less than 0.05

the exact binomial test

- the **exact binomial test** is used when comparing an estimated probability (e.g. the accuracy) to some fixed value
  - 40 correct guesses out of 50
  - is the true accuracy really different from 0.70?
- if the null hypothesis is true, then this experiment corresponds to a binomially distributed r.v. with parameters 50 and 0.70
- we compute the p-value as the probability of getting an outcome at least as unusual as 40

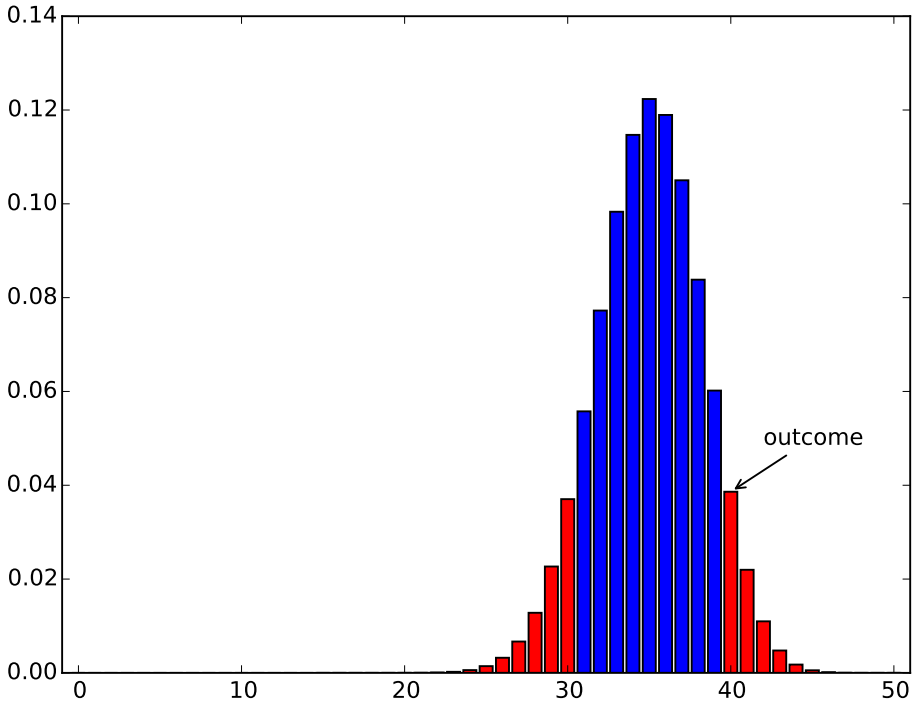
example

- 40 correct guesses out of 50
- is the true accuracy really different from 0.70?



example

- 40 correct guesses out of 50
- is the true accuracy really different from 0.70?



- the p-value is 0.16, so it seems we can't be sure!

implementing the exact binomial test in Scipy

- assume we made `x` correct guesses out of `n`
- is the accuracy significantly different from `test_acc`?
- the *p*-value is the sum of the probabilities of the outcomes that are at least as "unusual" as `x`:

```
import scipy.stats

def exact_binom_test(x, n, test_acc):
    rv = scipy.stats.binom(n, test_acc)
    p_x = rv.pmf(x)
    p_value = 0
    for i in range(0, n+1):
        p_i = rv.pmf(i)
        if p_i <= p_x:
            p_value += p_i
    return p_value
```

- actually, we don't have to implement it since there is a function `scipy.stats.binom_test` that does exactly this!

overview

overview of the second programming assignment

significance testing for the accuracy

comparing two classifiers

comparing two classifiers

- I'm comparing a Naive Bayes and a perceptron classifier
- we evaluate them on the **same test set**
- the NB classifier had 186 correct out of 312 guesses
- ...and the perceptron had 164 correct guesses
- so the ML estimates of the accuracies are 0.60 and 0.53, respectively
- but can I be certain that the NB classifier is really better?

contingency table

- we make a table that compares the errors of the two classifiers:

	NB correct	NB incorrect
perc correct	A = 125	**B = 39**
perc incorrect	**C = 61**	D = 87

- if NB is about as good as the perceptron, the B and C values should be similar
  - conversely if they are really different, B and C should differ
- are these B and C value **unusual**?

McNemar's test

- in **McNemar's test**, we model the discrepancies (the B and C values)

	NB correct	NB incorrect
perc correct	A = 125	**B = 39**
perc incorrect	**C = 61**	D = 87

- there are a number of variants of this test
  - the original formulation:
    Quinn McNemar (1947). *Note on the sampling error of the difference between correlated proportions or percentages*, Psychometrika 12:153-157.
- our version builds on the exact binomial test that we saw before

McNemar's test (continued)

	NB correct	NB incorrect
perc correct	A = 125	**B = 39**
perc incorrect	**C = 61**	D = 87

- the number of discrepancies is B+C
- how are the discrepancies distributed?
  - if the two systems are equivalent, the discrepancies should be more or less evenly spread into the B and C boxes
  - it can be shown that B would be a binomial random variable with parameters B+C and 0.5
- so we can find the *p*-value (the "unusualness") like this:

```
p_value = scipy.stats.binom_test(B, B+C, 0.5)
```

- in this case it is 0.035, so it seems that NB is really better

next weeks

- lecture next Tuesday in L308:
  - HMM tagging and Viterbi recap
  - confidence intervals and comparisons for taggers
- next assignment: implementing a HMM tagger