# Maximum entropy assignment

Jean Mark Gawron

Linguistics 581
San Diego State University

March 2, 2017

# Outline

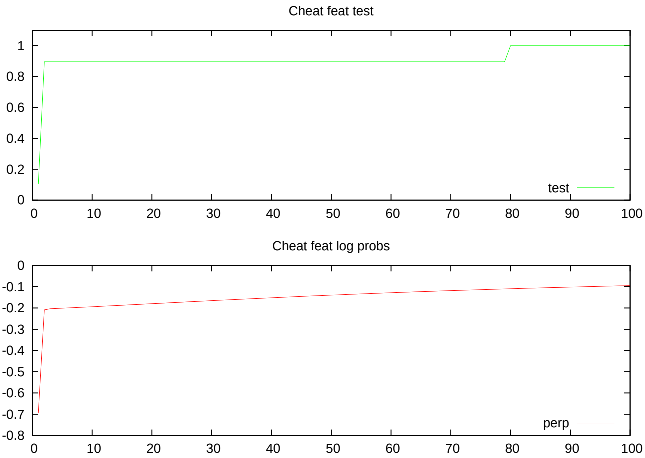
1. Algorithm used for model estimation is GIS (Generalized iterative scaling).
2. A model that always guesses HARD1 gets 80% (the ‘no feat’ model):

| | | |
|---|---|---|
| *HARD1* | *3455* | *0.797* |
| *HARD2* | *502* | *0.116* |
| *HARD3* | *376* | *0.087* |

3. We begin with the ‘cheat feat’ model

| | |
|---|---|
| *HARD1* | *True whenever the correct sense is HARD1* |
| *HARD3* | *True whenever the correct sense is HARD3* |

# Log Likelihood versus test results



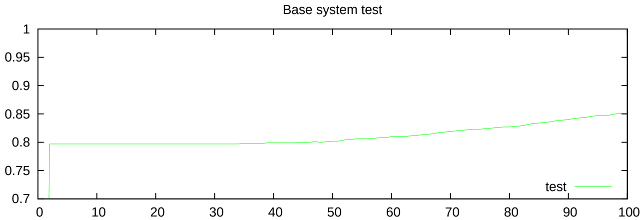

1.00
Stuck at .90
till iter # 80

# Your output

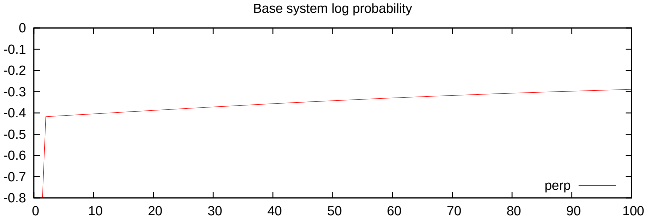
```
bulba> python -i call_maxent.py senseval-hard.evt 100
Reading data...
  Senses: HARD1 HARD3
Splitting into test & train...
Training classifier...
  ==> Training (100 iterations)

      Iteration    Log Likelihood    Accuracy
      ---------------------------------------
             1          -0.69315        0.104
             2          -0.20886        0.896
             3          -0.20371        0.896
             4          -0.20233        0.896
             5          -0.20101        0.896
             6          -0.19968        0.896
             7          -0.19834        0.896
             8          -0.19697        0.896
                    . . .
            78          -0.11112        0.896
            79          -0.11028        0.896
            80          -0.10944        1.000
                     . . .
            98          -0.09600        1.000
            99          -0.09534        1.000
           100          -0.09468        1.000
Testing classifier...
Accuracy: 1.0000
Total: 386
```

# Baseline system




0.85
Stuck at .80
till iter # 40



# Baseline evaluation

```
Testing classifier...
Accuracy: 0.8732
Total: 410

Label                  Precision      Recall
---------------------------------------------
HARD1                      0.869       1.000
HARD2                      0.909       0.303
HARD3                      1.000       0.275

Label                   Num Corr
HARD1                 337
HARD2                 10
HARD3                 11
```

# What is optimized

```
        Iter          Log Likelihood     Score
-----------------------------------------------------------
                 ....
         25           -0.36695         0.793
         26           -0.36471         0.793
         27           -0.36251         0.793
         28           -0.36033         0.794
         29           -0.35819         0.793
         30           -0.35608         0.794
         31           -0.35400         0.795
                ....
```

# Summarizing

1. Log likelihood is guaranteed to increase every iteration, but accuracy is not. Accuracy may remain pegged to one value for many iterations, and then suddenly change.
2. Therefore, be patient. For a system with $n$ features, wait at least $n$ iterations [but remember cheat_feats!]
3. If log likelihood goes down, look for a bug.
4. If score consistently goes down, or remains fixed, look for new features.

# Contextual predicates/Features

## Contextual predicates based on vocabulary

| | |
|---|---|
| Baseline | 100 most frequent vocab items |
| Hand | 100 most frequent + hand selected |
| Optimal | 100 features selected by feature selection algorithm |

## Features

Every class cls, contextual predicate cp pair defines a feature:

$$\mathsf{F}_{\mathsf{cp,cls}}(\kappa, c) = 1 \text{ iff } \mathsf{cp}(\kappa) \text{ is true and } c = cls$$

Example: given training example *The cushion is hard*, classified HARD3,

$$\mathsf{F}_{\text{cushion,HARD3}}(\textit{The cushion is hard}, \text{HARD3}) = 1$$

# Adding features by hand

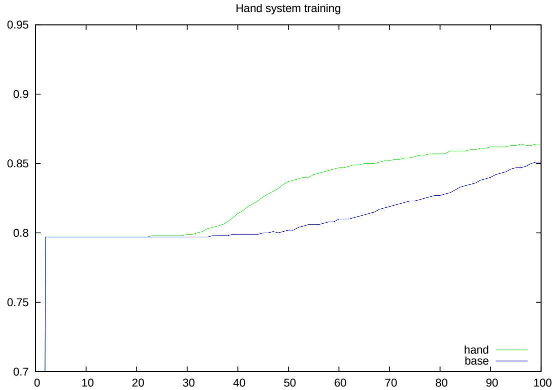
In *extract_vocab*, add a vocab item plus a fake count:

```
vocab[cushion_NN] = 1000
vocab[soft_JJ] = 1000
vocab[cover_NN] = 1000
...
```

Running *call_max_ent.py* always shows 20 most informative features:
classifier.show_most_informative_features(n=20)

```
 ....
1.057 wood_NN==True and label is HARD3
1.026 seat_NN==True and label is HARD3
....
```

# Hand vocab system




Hand .864 Base .851
+ 14 Nouns

# Increasing feats and training iterations

**A score $> 99\%$**

| | Num iter | Vocab size | | Error | Log Likelihood |
|---|---|---|---|---|---|
| a. | 300 | 100 | Training | 0.12863 | 1.44553 |
| | | | Test | 0.15765 | 1.50773 |
| b. | 300 | 1000 | Training | 0.05554 | 1.22309 |
| | | | Test | 0.11059 | 1.37440 |
| c. | 300 | 2000 | Training | 0.03022 | 1.16815 |
| | | | Test | 0.12000 | 1.40761 |
| d. | 500 | 2000 | Training | 0.00748 | 1.05689 |
| | | | Test | 0.12471 | 1.50585 |

# What is overtraining?

1. A very high level of performance on the training set linked with poor performance on tests (unseen data).
2. Features that don't carry information about significant reproducible aspects of the problem you're trying to solve still carry information about accidental properties of the training set. [the fact that there's a hyphen in some example with sense HARD2]
3. Features that do carry information about significant reproducible aspects of the problem can be overvalued or undervalued because they correlate positively or negatively with accidental properties of the training set.
4. The information these uninformative features or wrongly valued features are representing is **noise** (with respect to **our** problem)
5. In overtraining, the model is learning noise. In effect, it is memorizing the training set.

# Causes of overtraining

1. Class of models too powerful.

   *Linear models versus K Nearest Neighbors (KNN)*

2. Too many features: take away some features and the model does better.
3. The wrong features: Will look just like too many features, except that taking away features never makes things better.

# Feature selection: the problem

*Earlier we divided the statistical modeling problem into two steps: finding appropriate facts about the data, and incorporating these facts into the model. Up to this point we have proceeded by assuming that the first task was somehow performed for us. Even in the simple example of Section 2, we did not explicitly state how we selected those particular constraints. That is, why is the fact that [French]* dans *or* à *was chosen by the expert translator 50% of the time [as the translation of* in*] any more important than countless other facts contained in the data? Berger et al. (1996), p. 46*

# A separate problem

*In fact, the principle of maximum entropy does not directly concern itself with the issue of feature selection, it merely provides a recipe for combining constraints into a model. But the feature selection problem is critical, since the universe of possible constraints is typically in the thousands or even millions.*
*Berger et al. (1996), p. 46*

# Consequence

### When Python becomes impractical

Choosing model features means running model estimation algorithms countless times on *held out data* (data you're not using for training). In order to do that each run has to be fast. So efficiency becomes a research consideration. At this point the fact that youre using a language like Python (instead of C) becomes important.

# Feature selection algorithm

1. Start with no feature model (always guesses most likely sense).
2. Iterate through all possible feats, estimating the log likelihood gain. using Newton's method to compute a one-dimensional approximation.
3. Add the feature with the maximum likelihood gain.
4. Re-estimate the model. Repeat from step 2.

# Stopping criterion

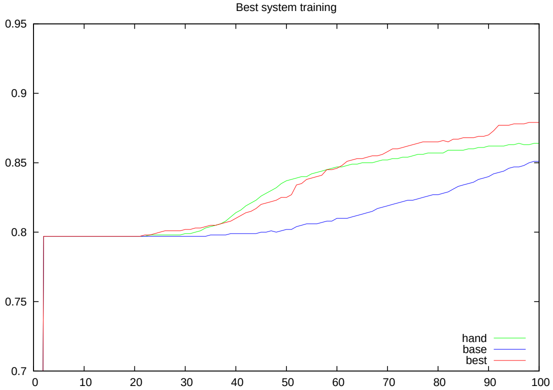
### Held-out data [dev training/test]

Test the model on held out data. When the log likelihood goes down, you have evidence of overtraining. Stop adding features. [Probably a little too strict.]

# Problems being solved

1. Discovery: What aspects of the event carry information relevant to our classification problem.
2. Alleviating overtraining problems. Our stopping criterion tries to determine when overtraining has started.

# Selected feat system




| | Best | Hand | Base |
|---|---|---|---|
| Train | 0.88 | 0.86 | 0.85 |
| Test | 0.91 | 0.88 | 0.87 |

# References I

Berger, Adam, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics* 22(1):39–71.

# Some of the selected features

[missing now]