Lecture 3
Classification

CS 585, Fall 2015
Introduction to Natural Language Processing
http://people.cs.umass.edu/~brenocon/inlp2015/

Brendan O'Connor

Your TA: Ari Kobren

- http://people.cs.umass.edu/~akobren/

- Python demo
- Overfitting, pseudocounts
- Intro: logistic regression

Multinomial Naive Bayes

$$P(y \mid w_1..w_T) \propto P(y) \prod_t P(w_t \mid y)$$

Tokens in doc

Parameters: $P(w \mid y)$ for each document category **y** and wordtype ***w***

$P(y)$ prior distribution over document categories **y**

<u>*Learning*</u>: Estimate parameters as **pseudocounted** frequency ratios

$$P(w \mid y, \alpha) = \frac{\#(w \text{ occurrences in docs with label } y) + \alpha}{\#(\text{tokens total across docs with label } y) + V\alpha}$$

<u>*Predictions*</u>:

Predict class $\arg\max_y P(Y = y \mid w_1..w_T)$

or, predict prob of classes...

Why Pseudocounts?

$$P(y \mid w_1..w_T) \propto P(y) \prod_t P(w_t \mid y)$$

Learning: Estimate parameters as **pseudocounted** frequency ratios

$$P(w \mid y, \alpha) = \frac{\#(w \text{ occurrences in docs with label } y) + \alpha}{\#(\text{tokens total across docs with label } y) + V\alpha}$$

- alpha = 0 => ?
- alpha = 0.000001 ==> ?

Why Pseudocounts?

$$P(y \mid w_1..w_T) \propto P(y) \prod_t P(w_t \mid y)$$

<u>*Learning*</u>: Estimate parameters as **pseudocounted** frequency ratios

$$P(w \mid y, \alpha) = \frac{\#(w \text{ occurrences in docs with label } y) + \alpha}{\#(\text{tokens total across docs with label } y) + V\alpha}$$

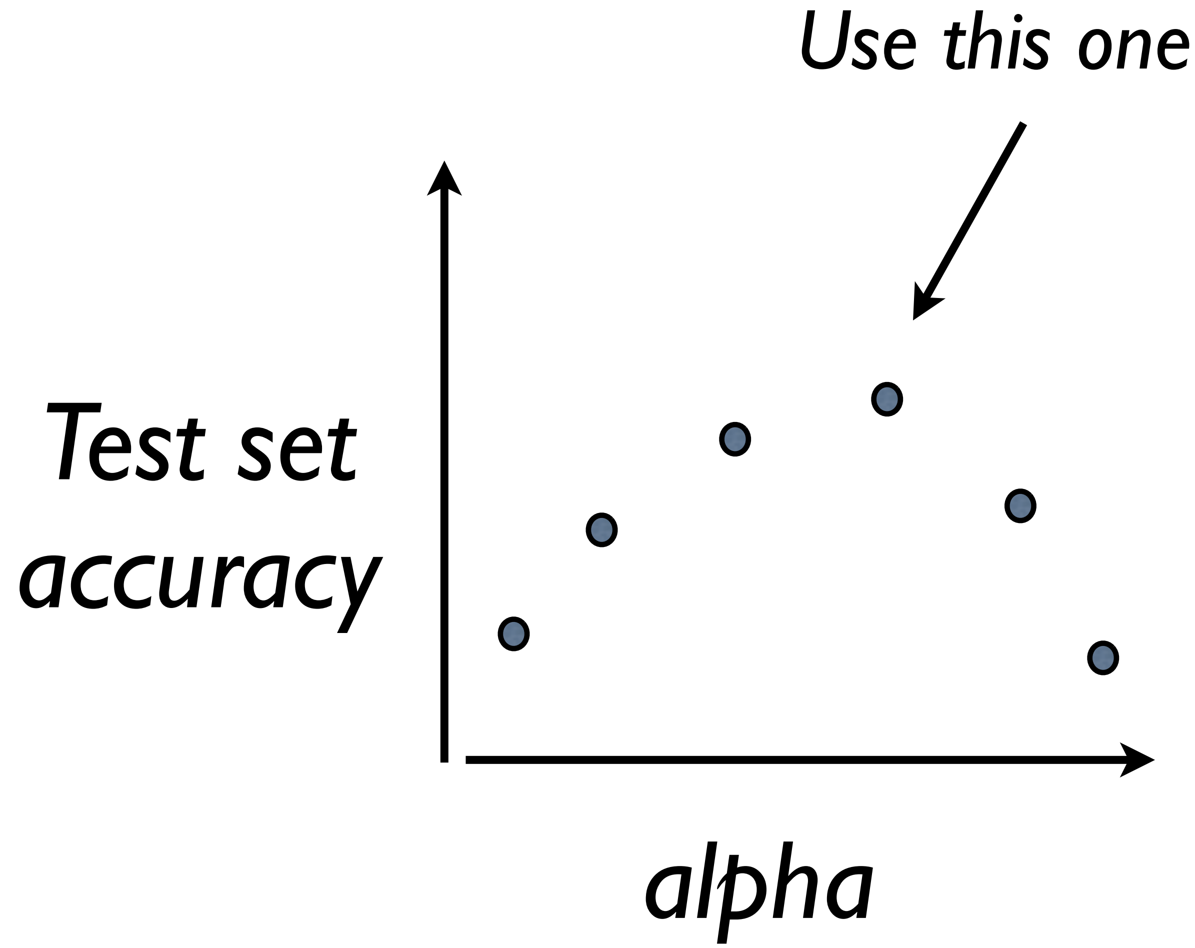
- alpha = 0 => ?
- alpha = 0.000001 ==> ?
- QUESTION: as alpha gets higher, posterior probabilities tend to
  - (A) 50%
  - (B) P(y)
  - (C) 100%
  - (D) No common trend

Overfitting

- Overfitting: model cares too much about training data
- To check: held-out data, e.g. Train vs Test
  - Training vs test accuracy: which is higher?
- pseudocount parameter combats overfitting

How to set the pseudocount?

- Split data into train versus test.
- Try different pseudocounts. For each train the model and predict on the held-out data. Choose lambda that does best on test set: e.g. maximizes accuracy or likelihood.
- What values to try? Often we use a grid search
  - e.g. (2^-2, 2^-1 ... 2^4, 2^5)



Hopefully looks like this

Data splitting

- Train vs Test
- Better: use
  - Train: for fitting model *<u>parameters</u>*
  - Dev: for tuning *<u>hyperparameters</u>*
  - Test: reserve for final evaluation
- Cross-validation

Feature engineering

- What's your word/feature representation?
  - tokenization rules: splitting on whitespace?
  - lowercase same as uppercase?
  - numbers?
  - punctuation?
  - phrases?