Natural Language Processing

Lecture 13—10/6/2015

Jim Martin

Today

- Multinomial Logistic Regression
  - Aka log-linear models or maximum entropy (maxent)
  - Components of the model
  - Learning the parameters

Logistic Regression Models

- Estimate P(c|d) directly (discriminative model)
  - Features
  - Model w/ weights on features
  - Classification

Features

- The kind of features used in NLP-oriented machine learning systems typically involve
  - Binary values
    - Think of a feature as being on or off rather than as a feature with a value
  - Values that are relative to an object/class pair rather than being a function of the object alone.
  - Typically have lots and lots of features (100,000s of features isn't unusual.)

POS Features

$$f_3(c,x) = \begin{cases} 1 & \text{if suffix}(word_i) = \text{"ing"} \ \& \ c = \text{VBG} \\ 0 & \text{otherwise} \end{cases}$$

$$f_4(c,x) = \begin{cases} 1 & \text{if is\_lower\_case}(word_i) \ \& \ c = \text{VB} \\ 0 & \text{otherwise} \end{cases}$$

Sentiment Features

$$f_1(c,x) = \begin{cases} 1 & \text{if "great"} \in x \ \& \ c = + \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(c,x) = \begin{cases} 1 & \text{if "second-rate"} \in x \ \& \ c = - \\ 0 & \text{otherwise} \end{cases}$$

$$f_3(c,x) = \begin{cases} 1 & \text{if "no"} \in x \ \& \ c = - \\ 0 & \text{otherwise} \end{cases}$$

$$f_4(c,x) = \begin{cases} 1 & \text{if "enjoy"} \in x \ \& \ c = - \\ 0 & \text{otherwise} \end{cases}$$

Sentiment Features w/ Weights

$$1.9 \quad f_1(c,x) = \begin{cases} 1 & \text{if "great"} \in x \;\&\; c = + \\ 0 & \text{otherwise} \end{cases}$$

$$.9 \quad f_2(c,x) = \begin{cases} 1 & \text{if "second-rate"} \in x \;\&\; c = - \\ 0 & \text{otherwise} \end{cases}$$

$$.7 \quad f_3(c,x) = \begin{cases} 1 & \text{if "no"} \in x \;\&\; c = - \\ 0 & \text{otherwise} \end{cases}$$

$$-.8 \quad f_4(c,x) = \begin{cases} 1 & \text{if "enjoy"} \in x \;\&\; c = - \\ 0 & \text{otherwise} \end{cases}$$

Logistic Regression Model

$$p(c|x) = \frac{\exp\left(\sum_{i=1}^{N} w_i f_i(c,x)\right)}{\sum_{c' \in C} \exp\left(\sum_{i=1}^{N} w_i f_i(c',x)\right)}$$

Logistic Regression Model

– 0.7   -0.8 –

... there are virtually no surprises, and the writing is second-rate. So why did I enjoy it so much? For one thing, the cast is great.

– .9   1.9 +

$$f_1(c,x) = \begin{cases} 1 & \text{if "great"} \in x \;\&\; c = + \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(c,x) = \begin{cases} 1 & \text{if "second-rate"} \in x \;\&\; c = - \\ 0 & \text{otherwise} \end{cases}$$

$$f_3(c,x) = \begin{cases} 1 & \text{if "no"} \in x \;\&\; c = - \\ 0 & \text{otherwise} \end{cases}$$

$$f_4(c,x) = \begin{cases} 1 & \text{if "enjoy"} \in x \;\&\; c = - \\ 0 & \text{otherwise} \end{cases}$$

$$p(c|x) = \frac{\exp\left(\sum_{i=1}^{N} w_i f_i(c,x)\right)}{\sum_{c' \in C} \exp\left(\sum_{i=1}^{N} w_i f_i(c',x)\right)}$$

Logistic Regression Model

– 0.7   -0.8 –

... there are virtually **no** surprises, and the writing is **second-rate**. So why did I **enjoy** it so much? For one thing, the cast is **great**.

.9 –   1.9 +

$$f_1(c,x) = \begin{cases} 1 & \text{if "great"} \in x \;\&\; c = + \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(c,x) = \begin{cases} 1 & \text{if "second-rate"} \in x \;\& \\ 0 & \text{otherwise} \end{cases}$$

$$f_3(c,x) = \begin{cases} 1 & \text{if "no"} \in x \;\&\; c = - \\ 0 & \text{otherwise} \end{cases}$$

$$f_4(c,x) = \begin{cases} 1 & \text{if "enjoy"} \in x \;\&\; c = \\ 0 & \text{otherwise} \end{cases}$$

$$P(+|x) = \frac{e^{1.9}}{e^{1.9} + e^{.9}e^{.7}e^{-.8}} = .82$$

$$P(-|x) = \frac{e^{.9}e^{.7}e^{-.8}}{e^{1.9} + e^{.9}e^{.7}e^{-.8}} = .18$$

Argmax

- If we didn't really care about the probabilities then this

$$p(c|x) = \frac{\exp\left(\sum_{i=1}^{N} w_i f_i(c,x)\right)}{\sum_{c' \in C} \exp\left(\sum_{i=1}^{N} w_i f_i(c',x)\right)}$$

- Reduces to the just a comparison of the sum of the weights

1.9 > .9+.7+-.8

1.9 > .8

Model

- So back to P(C|D)… given:
  - A training set of documents (D)
  - And an associated set of labels (C)
  - A finite set of features of the documents and classes
  - And a weight vector over the features (w)
  - This produces a proper probability distribution over the classes given an document

$$p(c|x) = \frac{\exp\left(\sum_{i=1}^{N} w_i f_i(c,x)\right)}{\sum_{c' \in C} \exp\left(\sum_{i=1}^{N} w_i f_i(c',x)\right)}$$

Learning the Weights

- We have a training set of labels and documents (y, x)
- We're going to use a Maximum Likelihood approach…
  - Choose the parameters (weights) that maximize the probability of the labels given the observations (features derived from the documents)
  - And we'll do that in log-space, so maximize the log prob of the labels given the data

Learning the Weights

- So choose the weights that maximizes

$$L(w) = \sum_j \log P(y^{(j)}|x^{(j)})$$

Learning the Weights

- So choose the weights that maximizes

$$= log \sum_j \frac{\exp\left(\sum_{i=1}^{N} w_i f_i(y^{(j)}, x^{(j)})\right)}{\sum_{y' \in Y} \exp\left(\sum_{i=1}^{N} w_i f_i(y'^{(j)}, x^{(j)})\right)}$$

Learning the Weights

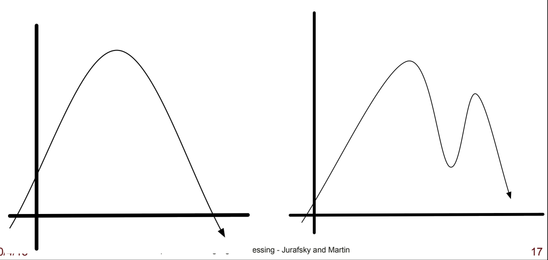
- So choose the weights that maximizes the following in log space

$$\log \sum_j \exp\left(\sum_{i=1}^{N} w_i f_i(y^{(j)}, x^{(j)})\right) - \log \sum_j \sum_{y' \in Y} \exp\left(\sum_{i=1}^{N} w_i f_i(y'^{(j)}, x^{(j)})\right)$$

- Fortunately, that corresponds neatly to a convex optimization problem for which there are many possible approaches

Convex Problems

- What does convex/concave mean?
  - And why does it matter



Derivatives

$$L'(w) = \sum_j f_k(y^{(j)}, x^{(j)}) - \sum_j \sum_{y' \in Y} P(y'|x^{(j)}) f_k(y'^{(j)}, x^{(j)})$$

$$L'(w) = \sum_j \text{Observed count}(f_k) - \text{Expected count}(f_k)$$

Gradient Ascent

- Initialize the weights to zero
  - *w* = 0
- Loop until convergence
  - Calculate derivative for each feature
    - δ_k for each of the k features

$$L'(w) = \sum_j \text{Observed count}(f_k) - \text{Expected count}(f_k)$$

    - Set w = w + $\beta * \delta$ for each feature k

Optimization

- In practice, that can actually be slow to converge because the algorithm can either be taking steps
  - That are too small and hence take us too long to get where we're going
  - Or too large which leads us to overshoot the target and wander around too much
- Fortunately, you don't have to worry about this. Lots of packages available where you need to specify L and L' and you're done.
  - LBFGs

Overfitting

- A problem with the approach as we've described it is that it is overly eager to match the training set as closely as it can
- With thousands or millions of features this can lead to poor performance on new data
- With logistic models this usually manifests itself as extremely large weights on a limited set of features

7

Regularization

- Solution is to add a "penalty term" to the objective function.
- The job of the penalty term is to squash the weights of features that are getting out of control.

$$\hat{w} = \operatorname*{argmax}_{w} \sum_{j} \log P(y^{(j)}|x^{(j)}) - \alpha R(w)$$

Parameter > 0 learned from held out dev data.

Regularization

- Solution is to add a "penalty term" to the objective function.
- The job of the penalty term is to squash the weights of features that are getting out of control.

$$\hat{w} = \operatorname*{argmax}_{w} \sum_{j} \log P(y^{(j)}|x^{(j)}) - \alpha R(w)$$

$$R(W) = ||W||_2^2 = \sum_{j=1}^{N} w_j^2$$

Regularization and Learning

- With regularization, the parameter learning has to balance finding models that match the training data well and use small weights.

Review

- Finite state methods
- Practical issues in segmentation and tokenization
- N-gram language models
- HMMs
  - HMMs applied to POS tagging
- Logistic regression and text classification