

A Virtual Manipulative for Learning Log-Linear Models

Author 1

XYZ Company
111 Anywhere Street
Mytown, NY 10000, USA
author1@xyz.org

Author 2

XYZ Company
111 Anywhere Street
Mytown, NY 10000, USA
author1@xyz.org

Abstract

We present a virtual manipulative for regularized conditional log-linear models.

1 Introduction

except for reading of data files, purely client-side \Rightarrow very easy to set-up; open-source; data input format makes it extensible; individual lessons can be tailored (e.g., hide/show buttons, different tool-tips for lessons)

Why do we focus on shapes, rather than words?

[FF: maybe we can argue via virtual manipulatives]

[FF: what is the history of virtual manipulatives in teaching CS? NLP?] the HMM spreadsheet (Eisner, 2002)

2 Regularized Conditional Log-Linear Models

Our aim is to provide an intuitive understanding of regularized conditional log-linear models. Given K features f_k representing N data points $\{(x_i, y_i)\}_{i=1}^N$, we are interested in estimating distributions

$$\hat{p}_{\vec{\theta}}(y \mid x) = \frac{u(x, y)}{\sum_{y'} u(x, y')}, \quad (1)$$

where $u(x, y)$ represents the unnormalized probability

$$u(x, y) = \exp(\vec{\theta} \cdot \vec{f}(x, y)) \quad (2)$$

$$= \exp\left(\sum_{k=1}^K \theta_k f_k(x, y)\right). \quad (3)$$

As our model $\hat{p}_{\vec{\theta}}$ is fully described by the feature weights $\vec{\theta}$, we find the weights that maximize the

regularized conditional log-likelihood (4):

$$F(\vec{\theta}) = \sum_{i=1}^N \log \hat{p}_{\vec{\theta}}(y_i \mid x_i) - C \cdot R(\vec{\theta}). \quad (4)$$

Here, $C \geq 0$ is the regularization penalty for the regularizer $R(\vec{\theta})$. We will generally refer to the **full model** as that of (1) with objective (4).

However, because the full model may present too many subtleties to be grasped all at once, we also consider “simplified” global models $\hat{p}_{\vec{\theta}_{\text{global}}}(y)$. These context-nescent models allow students to focus initially on understanding some of the core underlying concepts of exponential models, such as feature interactions and weight tradeoffs, before progressing to conditioned models.

[FF: move to a later section] Using \tilde{p} to represent the empirical distribution, the gradient of (4) is, in general,

$$\nabla_{\vec{\theta}} F = \mathbb{E}_{\tilde{p}}[\vec{f}(x, y)] - \mathbb{E}_{\hat{p}}[\vec{f}(x, y)] - C \nabla_{\vec{\theta}} R(\vec{\theta}). \quad (5)$$

We consider models where $R(\vec{\theta}) = \vec{0}$ (no regularization), $R(\vec{\theta}) = \|\vec{\theta}\|_1$, and $R(\vec{\theta}) = \|\vec{\theta}\|_2^2$: we special-case the optimizer to handle the non-differentiable ℓ_1 regularization, thus providing a friendly educational environment in which the student may explore the differences between ℓ_1 and ℓ_2 regularization.

3 Our Notes

[FF: These are simply copied from the Google doc titled “600.465: Maxent Notes.” This section could be retitled general pedagogical aims, or something of the sort.]

- If the striped feature is predicted to occur less

often than it actually does, you should raise its weight.

- Its possible to overfit the training data. Regularization compensates for that and can in fact make you underfit.
 - In particular, weights may zoom off to +infinity or -infinity if a feature is always or never present on the *observed* examples (may need to cook special datasets for this)
- Interactions:
 - Raising one weight may reduce or reverse the need to raise other weights. This can be seen by watching the gradient as we slide the slider.
 - Can share features across conditions and this helps regularizer even if likelihood is the same
 - Features that only fire on conditions have no effect on conditional distribution
 - Feature conjunctions: fewer vs. more features
 - Feature that everything/nothing has — weights go to $\pm\infty$
 - Opposing features, e.g., solid vs striped, where there are only 2 options (or, red vs. blue)
- Likelihood always goes up if you follow gradient
 - gradient = observed - expected count (-regularizer)
 - This is evident in the LL-bar at the top
- LL is maximized when you match the empirical (except for overfitting?)
- Frequent conditions more influential
- Some distributions can't be matched — but you get generalization
- The initial setting where all weights = 0 gives the uniform distribution.
 - Some further understanding of the entropy view? (See below.)

[FF: Make sure to address this:] We should note that we are *not* concerned with computational issues here, e.g., that of tractably computing the per-context normalization factor. While efficiently computing the normalization factor is a crucial component to practical log-linear models, our primary concern is to provide an intuitive understand.

[FF: work on the wording here...]

[FF: It's also important to talk about convexity of the objective.] That is, we'll always find a unique $\vec{\theta}^*$ that solves (4).

4 Virtual Manipulative Overview

See Figure 1 for a screenshot.

5 Usability

“New Counts” button The other use is to help the user experiment with datasets of different sizes, by changing N to scale the counts and then clicking “New counts.”

5.1 Data description

Required

- a set of contexts (if missing, will be assumed to contain only one context)
- a set of features; some of these may be marked as hidden
- for each context, a set of events and visual positions for them [FF: explain why this visual positioning is pedagogically important, i.e., aligning circles vs. triangles, and solid vs. striped can make feature contrasts stand out more]

Optional

- weights for some or all features (if missing, will be imputed from the prior $N(0, I)$ and the supplied count vectors) If no counts are supplied, then imputation is equivalent to simple sampling from $N(0, I)$. More generally, imputation requires MH sampling of the feature weights (and its wise to initialize the sampler to a MAP estimate found by the solver). We may not implement this case immediately, in which case the weights may stay unknown. In that case we have to gray out a couple of buttons and treat contexts without count vectors as if they had no observations.

Not done [FF: sadly, these didn't get tended to]

Lesson 1

Welcome! This interactive visualization will help you understand the popular technique of log-linear modeling.

Try it out: The sliders below control the parameters ("weights") of a log-linear model. When you increase the `circle` weight, which filled shapes get bigger? Which ones get smaller?

One game is to try to match all 4 shapes to the gray outlines. You will need to use both sliders. A shape will turn gray if it matches well. It turns **red** if it is too small, **blue** if it is too big. Note: You may like to zoom in with your browser.

What the picture means: Your model is defining a probability for each shape. When the weights are 0, all 4 shapes have equal probability of 1/4. This is shown by their equal areas.

Log Likelihood Scores

Current LL: -87.31

Type Counts: Observed and Expected

 $N =$

80

| | | | |
|---|---|----|---|
| 30 | 27 | 15 | 7 |
|  |  | | |
| 10 | 21 | 5 | 5 |
|  |  | | |

☒ Show gradient

Regularization

☐ None☐ ℓ_1 ☒ ℓ_2

C = 4

[Step](#)[Solve](#)

Step size = 0.1

circle

0.217317

solid

1.368204

Zero weights

Figure 1: Screenshot of the first lesson from the virtual manipulative.

- visual positions for all visible and hidden features (if missing, will be filled in heuristically)
- for each context, a total count N_x (if missing, will be imputed from an integerized gamma prior and the supplied event counts and the weights) In practice, we can forbid the lesson-maker from supplying only some of the event counts in a given context. In that case, either the whole count vector is given and N_x is just the sum, or none of the count vector is given and N_x is sampled from the gamma. for each (context, event) pair, a count (any missing counts for this context will be imputed) In practice, we can forbid the lesson-maker from supplying only some of the event counts in a given context. In that case, if any of this count vector is missing, then the whole thing is missing, and we can impute it simply by sampling N_x events from the distribution defined by the model weights.

References

Jason Eisner. 2002. An interactive spreadsheet for teaching the forward-backward algorithm. In Dragomir Radev and Chris Brew, editors, *Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching NLP and CL*, pages 10–18, Philadelphia, July.