Logistic regression (2)

Sharon Goldwater

29 October 2018
Recap: logistic regression

• A discriminative classifier: model \( P(y|x) \) only, no generative process

• \( P(y|x) \) depends on dot product of weights and features

• Avoids strong independence assumptions and can use arbitrary local/global features, so potentially better results than generative models

• But training is much more computationally intensive
Today’s lecture

- How do we train the weights? Practical issues to watch out for.
- Relationship to neural network models.
Training the model

Two ways to think about training:

- **What** is the goal of training (training objective)?
- **How** do we achieve that goal (training algorithm)?
Training generative models

• Easy to think in terms of how: counts/smoothing.

• But don’t forget the what:

<table>
<thead>
<tr>
<th>What</th>
<th>How</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximize the likelihood</td>
<td>take raw counts and normalize</td>
</tr>
<tr>
<td>Other objectives¹</td>
<td>use smoothed counts</td>
</tr>
</tbody>
</table>

¹Historically, smoothing methods were originally introduced purely as how: that is, without any particular justification as optimizing some objective function. However, as alluded to earlier, it was later discovered that many of these smoothing methods correspond to optimizing Bayesian objectives. So the what was discovered after the how.
Training logistic regression

Possible training objective:

- Given annotated data, choose weights that make the labels most probable under the model.

- That is, given items $x^{(1)} \ldots x^{(N)}$ with labels $y^{(1)} \ldots y^{(N)}$, choose

$$\hat{w} = \arg \max_{\vec{w}} \sum_{j} \log P(y^{(j)} | x^{(j)})$$

- This is conditional maximum likelihood estimation (CMLE).
Regularization

- Like MLE for generative models, CMLE can overfit training data.
  - For example, if some particular feature combination is only active for a single training example.

- So, add a regularization term to the equation
  - encourages weights closer to 0 unless lots of evidence otherwise.
  - various methods; see JM3 or ML texts for details (optional).

- In practice it may require some experimentation (dev set!) to choose which method and how strongly to penalize large weights.
Optimizing (regularized) cond. likelihood

• Unlike generative models, we can’t simply count and normalize.

• Instead, we use gradient-based methods, which iteratively update the weights.
  – Our objective is a function whose value depends on the weights.
  – So, compute the gradient (derivative) of the function with respect to the weights.
  – Update the weights to move toward the optimum of the objective function.
Visual intuition

• Changing $\vec{w}$ changes the value of the objective function.$^2$

• Follow the gradients to optimize the objective ("hill-climbing").

---

$^2$Here, we are maximizing an objective such as log prob. Using an objective such as negative log prob would require minimizing; in this case the objective function is also called a loss function.
But what if...?

- If there are multiple **local optima**, we won’t be guaranteed to find the **global optimum**.
Guarantees

• Luckily, (supervised) logistic regression does not have this problem.
  – With or without standard regularization, the objective has a single global optimum.
  – Good: results more reproducible, don’t depend on initialization.

• But it is worth worrying about in general!
  – Unsupervised learning often has this problem (eg for HMMs, PCFGs, and logistic regression); so do neural networks.
  – Bad: results may depend on initialization, can vary from run to run.
Logistic regression: summary

- model $P(y|x)$ only, have no generative process
- can use arbitrary local/global features, including correlated ones
- can use for classification, or choosing from n-best list.
- training involves iteratively updating the weights, so typically slower than for generative models (especially if very many features, or if time-consuming to extract).
- training objective has a single global optimum.

Similar ideas can be used for more complex models, e.g. sequence models for taggers that use spelling features.
Extension to neural network

- Logistic regression can be viewed as a building block of neural networks (a perceptron).

- Pictorially:
Extension to neural network

- Adding a **fully-connected layer** creates one of the simplest types of neural network: a **multi-layer perceptron (MLP)**.

\[
\frac{1}{Z} \exp\left(\sum \text{inputs}\right)
\]

\[
f\left(\sum \text{inputs}\right)
\]
Key features of MLP

- Contains one or more **hidden layers**
  - Each node applies a non-linear function to the sum of its inputs
  - Hidden layers can be viewed as learned representations (**embeddings**) of the input
  - Recall that **word embeddings** represent words as vectors, such that similar words have similar vectors.
  - (Actually, even basic logistic regression can produce word embeddings: see next week.)
Key features of MLP

- Contains one or more hidden layers
- A non-linear classifier: more powerful than logistic regression.
Key features of MLP

• Contains one or more hidden layers

• A non-linear classifier: more powerful than logistic regression.

• Also trained using gradient-based methods, but vulnerable to local optima, so can be more difficult to train.
Key features of MLP

- Contains one or more hidden layers
- A non-linear classifier: more powerful than logistic regression.
- Also trained using gradient-based methods, but vulnerable to local optima, so can be more difficult to train.
- Like other NNet architectures, really just a complex function computed by multiplying weights by inputs and passing through non-linearities. Not magic.