Log-linear Models (Review)

Instructor: Wei Xu

Many slides from Michael Collins
Overview

- Log-linear models
- Parameter estimation in log-linear models
- Smoothing/regularization in log-linear models
The General Problem

- We have some **input domain** $\mathcal{X}$
- Have a finite **label set** $\mathcal{Y}$
- Aim is to provide a **conditional probability** $p(y \mid x)$ for any $x, y$ where $x \in \mathcal{X}$, $y \in \mathcal{Y}$
Feature Vector Representations

- Aim is to provide a conditional probability $p(y \mid x)$ for “decision” $y$ given “history” $x$

- A feature is a function $f_k(x, y) \in \mathbb{R}$ (Often binary features or indicator functions $f_k(x, y) \in \{0, 1\}$).

- Say we have $m$ features $f_k$ for $k = 1 \ldots m$
  $\Rightarrow$ A feature vector $f(x, y) \in \mathbb{R}^m$ for any $x, y$

features are a property of both observation $x$ and the candidate output class $y$
Parameter Vectors

- Given features $f_k(x, y)$ for $k = 1 \ldots m$, also define a **parameter vector** $v \in \mathbb{R}^m$

- Each $(x, y)$ pair is then mapped to a “score”

$$v \cdot f(x, y) = \sum_k v_k f_k(x, y)$$

However, this doesn’t produce a legal probability.
Log-Linear Models

- We have some input domain $\mathcal{X}$, and a finite label set $\mathcal{Y}$. Aim is to provide a conditional probability $p(y \mid x)$ for any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.
- A feature is a function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ (Often binary features or indicator functions $f_k : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$).
- Say we have $m$ features $f_k$ for $k = 1 \ldots m$ ⇒ A feature vector $f(x, y) \in \mathbb{R}^m$ for any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.
- We also have a parameter vector $v \in \mathbb{R}^m$
- We define

$$p(y \mid x; v) = \frac{e^{v \cdot f(x, y)}}{\sum_{y' \in \mathcal{Y}} e^{v \cdot f(x, y')}}$$
Exercise
Why the name?

\[ \log p(y \mid x; v) = v \cdot f(x, y) - \log \sum_{y' \in \mathcal{Y}} e^{v \cdot f(x, y')} \]

- Linear term
- Normalization term
Overview

- Log-linear models
- Parameter estimation in log-linear models
- Smoothing/regularization in log-linear models
Maximum-Likelihood Estimation

- Maximum-likelihood estimates given training sample $(x^{(i)}, y^{(i)})$ for $i = 1 \ldots n$, each $(x^{(i)}, y^{(i)}) \in \mathcal{X} \times \mathcal{Y}$:

$$v_{ML} = \arg\max_{v \in \mathbb{R}^m} L(v)$$

where

$$L(v) = \sum_{i=1}^{n} \log p(y^{(i)} | x^{(i)}; v) = \sum_{i=1}^{n} v \cdot f(x^{(i)}, y^{(i)}) - \sum_{i=1}^{n} \log \sum_{y' \in \mathcal{Y}} e^{v \cdot f(x^{(i)}, y')}$$

concave function!
Calculating the Maximum-Likelihood Estimates

- Need to maximize:

\[
L(v) = \sum_{i=1}^{n} v \cdot f(x^{(i)}, y^{(i)}) - \sum_{i=1}^{n} \log \sum_{y' \in \mathcal{Y}} e^{v \cdot f(x^{(i)}, y')}
\]

- Calculating gradients:

\[
\frac{dL(v)}{dv_k} = \sum_{i=1}^{n} f_k(x^{(i)}, y^{(i)}) - \sum_{i=1}^{n} \frac{\sum_{y' \in \mathcal{Y}} f_k(x^{(i)}, y') e^{v \cdot f(x^{(i)}, y')}}{\sum_{z' \in \mathcal{Y}} e^{v \cdot f(x^{(i)}, z')}}
\]

\[
= \sum_{i=1}^{n} f_k(x^{(i)}, y^{(i)}) - \sum_{i=1}^{n} \sum_{y' \in \mathcal{Y}} f_k(x^{(i)}, y') \frac{e^{v \cdot f(x^{(i)}, y')}}{\sum_{z' \in \mathcal{Y}} e^{v \cdot f(x^{(i)}, z')}}
\]

= \sum_{i=1}^{n} f_k(x^{(i)}, y^{(i)}) - \underbrace{\sum_{i=1}^{n} \sum_{y' \in \mathcal{Y}} f_k(x^{(i)}, y') p(y' | x^{(i)}; v)}_{\text{Empirical counts}} - \underbrace{\sum_{i=1}^{n} \sum_{y' \in \mathcal{Y}} f_k(x^{(i)}, y') p(y' | x^{(i)}; v)}_{\text{Expected counts}}
Gradient Ascent Methods

- Need to maximize $L(v)$ where

$$\frac{dL(v)}{dv} = \sum_{i=1}^{n} f(x^{(i)}, y^{(i)}) - \sum_{i=1}^{n} \sum_{y' \in Y} f(x^{(i)}, y') p(y' | x^{(i)}; v)$$

Initialization: $v = 0$

Iterate until convergence:

- Calculate $\Delta = \frac{dL(v)}{dv}$
- Calculate $\beta_* = \arg \max_{\beta} L(v + \beta \Delta)$ (Line Search)
- Set $v \leftarrow v + \beta_* \Delta$
Conjugate Gradient Methods

- (Vanilla) gradient ascent can be very slow

- Conjugate gradient methods require calculation of gradient at each iteration, but do a line search in a direction which is a function of the current gradient, and the previous step taken.

- Conjugate gradient packages are widely available

In general: they require a function

\[
\text{calc\_gradient}(v) \rightarrow \left( L(v), \frac{dL(v)}{dv} \right)
\]

and that’s about it!

[https://www.youtube.com/watch?v=h4cG8jLGmKg](https://www.youtube.com/watch?v=h4cG8jLGmKg)

e.g. LBFGS Algorithm
(Limited-memory Broyden-Fletcher-Goldfarb-Shanno)
Overview

- Log-linear models
- Parameter estimation in log-linear models
- Smoothing/regularization in log-linear models
Smoothing in Log-Linear Models

- Say we have a feature:

\[ f_{100}(x, y) = \begin{cases} 
1 & \text{if current word } w_i \text{ is base and } y = Vt \\
0 & \text{otherwise}
\end{cases} \]

- In training data, base is seen 3 times, with Vt every time

- Maximum likelihood solution satisfies

\[
\sum_i f_{100}(x^{(i)}, y^{(i)}) = \sum_i \sum_y p(y \mid x^{(i)}; v) f_{100}(x^{(i)}, y)
\]

\[ \Rightarrow p(Vt \mid x^{(i)}; v) = 1 \text{ for any history } x^{(i)} \text{ where } w_i = \text{base} \]
\[ \Rightarrow v_{100} \to \infty \text{ at maximum-likelihood solution (most likely)} \]
\[ \Rightarrow p(Vt \mid x; v) = 1 \text{ for any test data history } x \text{ where } w = \text{base} \]
Regularization

- Modified loss function

\[ L(v) = \sum_{i=1}^{n} v \cdot f(x^{(i)}, y^{(i)}) - \sum_{i=1}^{n} \log \sum_{y' \in \mathcal{Y}} e^{v \cdot f(x^{(i)}, y')} - \frac{\lambda}{2} \sum_{k=1}^{m} v_k^2 \]

- Calculating gradients:

\[ \frac{dL(v)}{dv_k} = \sum_{i=1}^{n} f_k(x^{(i)}, y^{(i)}) - \sum_{i=1}^{n} \sum_{y' \in \mathcal{Y}} f_k(x^{(i)}, y') p(y' | x^{(i)}; v) - \lambda v_k \]

  \underbrace{\text{Empirical counts}}_{\text{Expected counts}}

- Can run conjugate gradient methods as before

- Adds a penalty for large weights