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 $k = 1 \dots m$
  $\Rightarrow$ 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** $v \in \mathbb{R}^m$

all possible m-dimensional real value vectors

- 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 $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$
(Often binary features or indicator functions $f_k : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\}$).
- 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 $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.
- We also have a **parameter vector** $v \in \mathbb{R}^m$
- We define

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

Softmax!

Exercise

Why the name?

$$\log p(y \mid x; v) = \underbrace{v \cdot f(x, y)}_{\text{Linear term}} - \underbrace{\log \sum_{y' \in \mathcal{Y}} e^{v \cdot f(x, y')}}_{\text{Normalization term}}$$

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 $(x^{(i)}, y^{(i)}) \in \mathcal{X} \times \mathcal{Y}$:

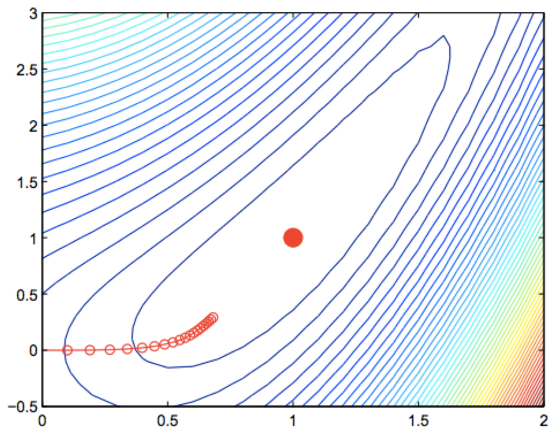
$$v_{ML} = \text{argmax}_{v \in \mathbb{R}^m} L(v)$$

where

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

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:

$$\begin{aligned}
\frac{dL(v)}{dv_k} &= \sum_{i=1}^{n} f_k(x^{(i)}, y^{(i)}) - \sum_{i=1}^{n} \frac{\sum_{y' \in \mathcal{Y}} f_k(x^{(i)}, y') e^{v \cdot f(x^{(i)}, y')}}{\sum_{z' \in \mathcal{Y}} e^{v \cdot f(x^{(i)}, z')}} \\
&= \sum_{i=1}^{n} f_k(x^{(i)}, y^{(i)}) - \sum_{i=1}^{n} \sum_{y' \in \mathcal{Y}} f_k(x^{(i)}, y') \frac{e^{v \cdot f(x^{(i)}, y')}}{\sum_{z' \in \mathcal{Y}} e^{v \cdot f(x^{(i)}, z')}} \\
&= \underbrace{\sum_{i=1}^{n} f_k(x^{(i)}, y^{(i)})}_{\text{Empirical counts}} - \underbrace{\sum_{i=1}^{n} \sum_{y' \in \mathcal{Y}} f_k(x^{(i)}, y') p(y' \mid x^{(i)}; v)}_{\text{Expected counts}}
\end{aligned}$$

Gradient Ascent Methods

- Need to maximize $L(v)$ where

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

Initialization: $v = 0$

Iterate until convergence:

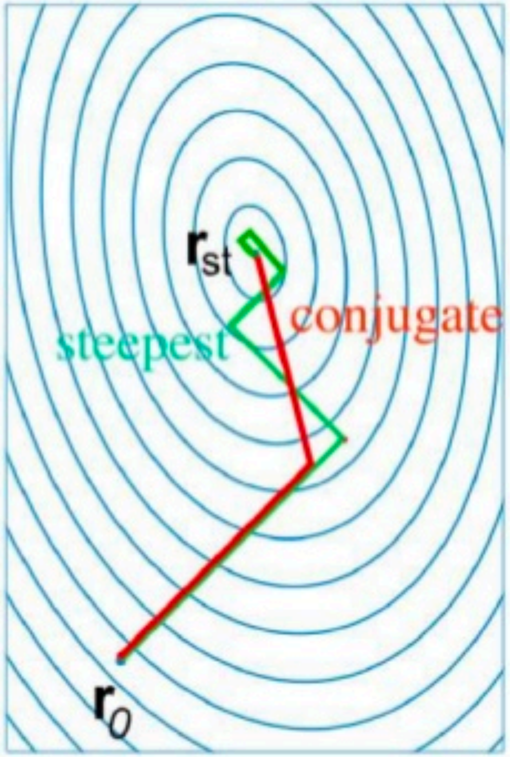
- Calculate $\Delta = \frac{dL(v)}{dv}$
- Calculate $\beta_* = \mathrm{argmax}_\beta L(v + \beta\Delta)$ (Line Search)
- Set $v \leftarrow v + \beta_*\Delta$

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

$$\texttt{calc\_gradient}(v) \rightarrow \left(L(v), \frac{dL(v)}{dv} \right)$$

and that's about it!



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

https://www.youtube.com/watch?v=h4cG8jLGmKg

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:

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

- In training data, base is seen 3 times, with Vt every time
- Maximum likelihood solution satisfies

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

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

$$\frac{dL(v)}{dv_k} = \underbrace{\sum_{i=1}^{n} f_k(x^{(i)}, y^{(i)})}_{\text{Empirical counts}} - \underbrace{\sum_{i=1}^{n} \sum_{y' \in \mathcal{Y}} f_k(x^{(i)}, y') p(y' \mid x^{(i)}; v)}_{\text{Expected counts}} - \lambda v_k$$

- Can run conjugate gradient methods as before
- Adds a penalty for large weights