

Lecture 11

Multinomial logistic regression

$$\vec{x} \in \mathbb{R}^D, \quad y \in \{1, \dots, C\}$$

So we define $(\vec{x}, 1)$ for each datapoint,
" $\vec{x}'$

we can create a single weight matrix:

$$\underbrace{\vec{a}}_{\text{dim} = C} = \underbrace{W \vec{x} + \vec{b}}_{\text{dim} = C} = \underbrace{\begin{pmatrix} W & \vec{b} \end{pmatrix}}_{\text{dim} = C \times (D+1)} \underbrace{\vec{x}'}_{\text{dim} = D+1}$$

$$\text{Recall that } p(y=c | \vec{x}, \vec{\theta}) = \frac{e^{a_c}}{\sum_{c'=1}^C e^{a_{c'}}} \quad (*)$$

" $(W, \vec{b})$

$$\text{Because } \sum_{c=1}^C p(y_n=c | \vec{x}_n, \vec{\theta}) = 1, \quad \forall n$$

we do not need the last row in $W' \Rightarrow$

$\Rightarrow$ we have $(C-1) \times (D+1)$ fitting prms.

We can use $p(y=c | \vec{x}, \vec{\theta}) = \frac{e^{a_c - a_C}}{1 + \sum_{c'=1}^{C-1} e^{a_{c'} - a_C}} \quad c=1, \dots, C-1$
Eq. (*) results in linear DBS $\Rightarrow$ can use
features $\vec{g}(\vec{x})$ to make non-linear DBS.

Objective function

$$- \mathcal{J}(\vec{\theta}) = - \frac{1}{N} \log \prod_{n=1}^N \prod_{c=1}^C \mu_{n,c}^{y_{n,c}} \quad \diamond$$

$$\begin{cases} y_{n,c} = \mathbb{I}(y_n=c) & \text{one-hot encoding} \\ \mu_{n,c} = \text{softmax}(\underbrace{\vec{a}_n}_= W \vec{x}_n + \vec{b})_c \end{cases}$$

$$\diamond - \frac{1}{N} \sum_{n=1}^N \sum_{c=1}^C y_{n,c} \log \mu_{n,c} = \frac{1}{N} \sum_{n=1}^N H(\vec{y}_n, \vec{\mu}_n),$$

where $H(\vec{p}, \vec{q}) = - \sum_{c=1}^C p_c \log q_c$.
cross-entropy

Optimization

First, consider $\frac{\partial \mu_c}{\partial a_j} = \frac{\partial}{\partial a_j} \left(\frac{e^{a_c}}{\sum_{c'=1}^C e^{a_{c'}}} \right) =$
 $= \frac{\delta_{cj} e^{a_c}}{\sum_{c'} e^{a_{c'}}} - \frac{e^{a_c}}{(\sum_{c'} e^{a_{c'}})^2} e^{a_j} = \mu_c (\delta_{cj} - \mu_j).$

Now, for a single datapoint $\vec{x}'$ (N=1) we have:

$$\frac{\partial}{\partial \vec{w}_j'} (-\mathcal{L}(\vec{\theta})) = - \sum_{c=1}^C \frac{y_{jc}}{\mu_c} \underbrace{\frac{\partial \mu_c}{\partial \vec{w}_j'}}_{\substack{\leftarrow \text{class label} \\ \vec{w}_j' = \text{vector of} \\ \text{dim } D+1}} \quad \text{where } a_j = \vec{w}_j' \cdot \vec{x}'$$

$$\frac{\partial \mu_c}{\partial a_j} \frac{\partial a_j}{\partial \vec{w}_j'} = \mu_c (\delta_{cj} - \mu_j) \vec{x}'$$

$$\ominus \sum_{c=1}^C y_{jc} (\mu_j - \delta_{cj}) \vec{x}' \underset{\substack{\uparrow \\ \sum_c y_c = 1}}{=} (\mu_j - y_j) \vec{x}'.$$

So, for all datapoints and all j (i.e., all classes), we obtain:

$$\underbrace{\vec{g}(\vec{\theta})}_{(D+1) \times (C-1) \text{ matrix}} = \frac{1}{N} \sum_{n=1}^N \vec{x}_n (\vec{\mu}_n - \vec{y}_n)^T$$

Likewise, we can compute the Hessian: (for a single datapoint $\vec{x}'$)

$$\frac{\partial^2}{\partial \vec{w}'_k \partial \vec{w}'_j} (-\mathcal{J}(\vec{\theta})) = \frac{\partial}{\partial \vec{w}'_k} [(\mu_j - y_j) \vec{x}'] =$$

$$= \mu_j (\delta_{jk} - \mu_k) \vec{x}' \vec{x}'^T.$$

Then for N datapoints we obtain:

$$\underbrace{H(\vec{\theta})}_{c, c'} = \frac{1}{N} \sum_{n=1}^N \mu_{n,c} (\delta_{cc'} - \mu_{n,c'}) \vec{x}_n \vec{x}_n^T$$

$(D+1) \times (D+1)$ matrix

The entire $H(\vec{\theta})$ is $(D+1)(C-1) \times (D+1)(C-1)$ and consists of $H_{c,c'}(\vec{\theta})$ submatrices.

One can use grad or Newton's optimization.

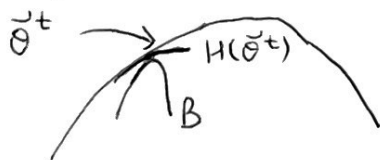
One can also use bound optimization:

$$\text{consider } \mathcal{J}(\vec{\theta}) = \mathcal{J}(\vec{\theta}^t) + (\vec{\theta} - \vec{\theta}^t) \cdot \vec{g}(\vec{\theta}^t) +$$

$$+ \frac{1}{2} (\vec{\theta} - \vec{\theta}^t) \cdot \underbrace{B}_{H(\vec{\theta}^t)} \cdot (\vec{\theta} - \vec{\theta}^t) \geq$$

$$\geq \mathcal{J}(\vec{\theta}^t) + (\vec{\theta} - \vec{\theta}^t) \cdot \vec{g}(\vec{\theta}^t) + \frac{1}{2} (\vec{\theta} - \vec{\theta}^t) \cdot B \cdot (\vec{\theta} - \vec{\theta}^t),$$

where $H(\vec{\theta}^t) \geq B$.



B is neg. def.

In our case (consider $G=2$ for simplicity), $H(\vec{\theta}^t) = -\frac{1}{N} X^T S_t X$, where

$$S_t = \text{diag} [\mu_{1,t}(1-\mu_{1,t}), \dots, \mu_{N,t}(1-\mu_{N,t})].$$

Since $\mu_{i,t}(1-\mu_{i,t}) \leq \frac{1}{4}$,

$B = -\frac{1}{4N} X^T X$ will be more
neg.-def. (i.e., it will curve
down faster)
than the 'real' curvature

Then we can use

$$\vec{\theta}^{t+1} = \vec{\theta}^t - 4N(X^T X)^{-1} \vec{g}(\vec{\theta}^t). \quad (**)$$

Since $(X^T X)^{-1}$ can be computed once,
Eq. (**) should be more efficient than
computing $N(X^T S_t X)^{-1}$ at each step.

MAP estimation

Recall that $\sum_{c=1}^G p(y_n = c | \vec{x}_n, W') = 1$ leads
to $(G-1)(D+1)$ indep. weights.

Now, let's use the original formula:

$$p(y=c | \vec{x}, W') = \frac{e^{d_c}}{\sum_{c'=1}^G e^{d_{c'}}}, \text{ which}$$

uses $G(D+1)$ weights.

Introduce ℓ_2 regularization:

$$-\mathcal{J}(W') = - \sum_{n=1}^N \log p(y_n | \vec{x}_n, W') + \frac{\lambda}{2} \sum_{c=1}^C \underbrace{\vec{w}_c^2}_{D+1 \text{ - dim vector}}$$

$(\lambda > 0)$

Recall that

$$\frac{\partial}{\partial w_{cj}} (-\mathcal{J}(W')) = \sum_{n=1}^N (\mu_{nc} - y_{nc}) x'_{nj}.$$

$\begin{matrix} \uparrow & \nwarrow \\ \text{class} & \text{feature} \\ \text{label} & \text{label} \\ c=1, \dots, C & j=1, \dots, D+1 \end{matrix}$

Now, we need

$$\frac{\partial}{\partial w_{cj}} (-\mathcal{J}(W')) + \lambda w_{cj} = 0, \text{ or}$$

$$\lambda w_{cj} = \sum_{n=1}^N (y_{nc} - \mu_{nc}) x'_{nj}.$$

But then

$$\begin{aligned} \lambda \sum_{c=1}^C w_{cj} &= \sum_{c,n} (y_{nc} - \mu_{nc}) x'_{nj} = \\ &= \sum_n \left[\underbrace{\sum_c y_{nc}}_{=1, \forall n} - \underbrace{\sum_c \mu_{nc}}_{=1, \forall n} \right] x'_{nj} = 0. \end{aligned}$$

Thus, $\sum_{c=1}^C w_{cj} = 0, \forall j$ at the minimum $\Rightarrow$ each ~~row~~ ^{column} in the W' matrix is constrained, we can use $G(D+1)$ weights.