Lecture 15

Multinomial logistic regression

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

If we define $\underbrace{(\vec{x}, 1)}_{=\vec{x}'}$ for each datapoint,

we can create a single weight matrix:

$$\underbrace{\vec{a}}_{\dim = C} = W\vec{x} + \vec{b} = \underbrace{\left(W \;\vdots\; \vec{b} \right)}_{=W'_{C\times(D+1)}} \underbrace{\vec{x}'}_{D+1 \text{ dim}}$$

Recall that $p(y=c \mid \vec{x}, \underbrace{\vec{\theta}}_{(W,\vec{b})}) = \dfrac{e^{a_c}}{\sum_{c'=1}^{C} e^{a_{c'}}}$ (*).

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

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

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

We can use $p(y=c \mid \vec{x}, \vec{\theta}) = \dfrac{e^{a_c - a_C}}{1 + \sum_{c'=1}^{C-1} e^{a_{c'} - a_C}}$, $c = 1, \dots, C-1$

Eq. (*) results in linear DBs $\Rightarrow$ can use features $\vec{\varphi}(\vec{x})$ to make non-linear DBs.

Objective function

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

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

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

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

cross-entropy

Optimization

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

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

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

$$\frac{\partial}{\partial \vec{w}_j'}\left(-\mathcal{L}(\vec{\theta})\right) = -\sum_{c=1}^{C}\frac{y_{\bullet c}}{\mu_c}\underbrace{\frac{\partial \mu_c}{\partial \vec{w}_j'}}_{=\frac{\partial \mu_c}{\partial a_j}\frac{\partial a_j}{\partial \vec{w}_j'}} = (=)$$

j ↖ class label

$\vec{w}_j'$ = vector of dim $D+1$

$a_j = \vec{w}_j' \cdot \vec{x}'$

$$= \mu_c(\delta_{cj} - \mu_j)\vec{x}'$$

$$(=) \sum_{c=1}^{C} y_c(\mu_j - \delta_{cj})\vec{x}' = (\mu_j - y_j)\vec{x}'.$$

($\sum_c y_c = 1$)

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 $\vec{x}'$)

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

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

Then for N datapoints we obtain:

$$\begin{cases} j \to c, \\ k \to c' \end{cases}$$

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

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

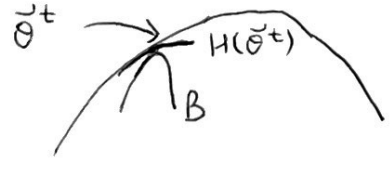
One can also use <u>bound optimization</u>:

consider $\mathcal{L}(\vec{\theta}) \simeq \mathcal{L}(\vec{\theta}^t) + (\vec{\theta}-\vec{\theta}^t)\cdot\vec{g}(\vec{\theta}^t) +$

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

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

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



B is neg. def.

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

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

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

Now, let's use the original formula:

$$p(y=c \mid \vec{x}, W') = \frac{e^{a_c}}{\sum_{c'=1}^{C} e^{a_{c'}}},$$ which uses $C(D+1)$ weights

Introduce l_2 regularization:

$$-\tilde{\mathcal{L}}(W') = -\sum_{n=1}^{N} \log p(y_n | \vec{x}_n, W') + \frac{\lambda}{2} \sum_{c=1}^{C} \vec{w}_c^{\,2}$$

$\lambda > 0$; $\vec{w}_c$: $D+1$ - dim vector

Recall that

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

c: class label, $c = 1, \ldots, C$; j: feature label, $j = 1, \ldots, D+1$

Now, we need

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

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

But then

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

$$= \sum_{n} \Big[\underbrace{\sum_{c} y_{nc}}_{\text{"}1, \forall n} - \underbrace{\sum_{c} \mu_{nc}}_{\text{"}1, \forall n} \Big] x'_{nj} = 0 .$$

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