— Solution notes —

Fourth week practicals in Machine learning 1 – 2024 – Paper 1

1 Multi-class Logistic Regression (September)

In the last lectures, we introduced the binary classification version of logistic regression. Here you will derive the gradients for the general case $K > 2$. Many of the preliminaries are in Bishop 4.3.4. For $K > 2$ the posterior probabilities take a generalized form of the sigmoid called softmax:

$$y_k = p(\mathcal{C}_k|\boldsymbol{\phi}) = \frac{\exp(a_k)}{\sum_i \exp(a_i)},$$

where $a_k = \mathbf{w}_k^T \boldsymbol{\phi}$ and $\boldsymbol{\phi}$ is short for $\boldsymbol{\phi}(\mathbf{x}) = (\phi_0(\mathbf{x}), \phi_1(\mathbf{x}), \ldots, \phi_{M-1}(\mathbf{x}))^T$ with $\phi_0(\mathbf{x}) = 1$. Note that the posterior for class k depends on all the other classes i; keep this in mind when working out the derivatives for $\mathbf{w}_k$. The training set is a pair of matrices $\boldsymbol{\Phi}$ and $\mathbf{T}$. Each row of $\mathbf{T}$ uses a one-hot encoding of the class labelling for that training example, meaning that the n-th row contains a row vector $\mathbf{t}_n^T$ with all entries zero except for the k-th entry which is equal to 1 if data point n belongs to class $\mathcal{C}_k$. Answer the following questions:

(a) Write down $\frac{\partial y_k}{\partial \mathbf{w}_j}$ (you have already calculated this derivative for the first assignment). Use the indicator function I_{kj} (or kronecker delta), which you can also think of as the element at position (k, j) of the identity matrix; which can be also denoted as $\mathbb{I}[k = j]$.

Answer:

The answer is obtained via the chain rule:

$$\frac{\partial y_k}{\partial \mathbf{w_j}} = \frac{\partial y_k}{\partial \mathbf{a}} \frac{\partial \mathbf{a}}{\partial \mathbf{w}_j} = \sum_p^K \frac{\partial y_k}{\partial a_p} \frac{\partial a_p}{\partial \mathbf{w}_j}. \tag{1}$$

The second part is directly derived as $\frac{\partial a_p}{\partial \mathbf{w}_j} = \boldsymbol{\phi}^T \delta_{pj}$. The first part is obtained using the quotient rule, which is given as follows. Let

$$f(x) = \frac{g(x)}{h(x)},$$

then

$$\frac{\partial f}{\partial x} = \frac{g'(x)h(x) - g(x)h'(x)}{h(x)^2},$$

where we use notation $g'(x) := \frac{\partial g}{\partial x}(x)$. We apply this to our problem by letting $f(x) = \frac{y_k}{\partial a_p}$, $g(a_1, \ldots, a_K) = \exp(a_k)$ and $h(a_1, \ldots, a_K) = \sum_i \exp(a_i)$. Then we have

$$g'(x) = \frac{\partial(\exp(a_k))}{\partial a_p} = \mathbb{I}[k = p] \exp(a_k),$$

which is non-zero when $k = p$ (and zero otherwise), and therefore, we can write it using the indicator function. We further have that

$$h'(x) = \exp(a_p).$$

To keep the derivation clean, let us denote the normalization factor with $C = \sum_i \exp(a_i)$ and recall that $y_k = \exp(a_k)/C$. Then expression $\frac{\partial y_k}{\partial a_p}$ becomes:

$$\begin{aligned}\frac{\partial y_k}{\partial a_p} &= \frac{\mathbb{I}[k=p]\exp(a_k)C - \exp(a_k)\exp(a_p)}{C^2} \\ &= \frac{\mathbb{I}[k=p]y_k C - y_k \exp(a_p)}{C} \\ &= \mathbb{I}[k=p]y_k - y_k y_p.\end{aligned}$$

We now complete (1) and conclude

$$\frac{\partial y_k}{\partial \mathbf{w_j}} = y_k(\boldsymbol{\phi})\boldsymbol{\phi}^T(\mathbb{I}[k=j] - y_j(\boldsymbol{\phi})).$$

(*b*) Write down the likelihood $p(\mathbf{T}|\mathbf{\Phi}, \mathbf{w}_1, \ldots, \mathbf{w}_K)$ as a product over N and K. Use the entries of $\mathbf{T}$ as selectors of the correct class. Then write down the log-likelihood $\log p(\mathbf{T}|\mathbf{\Phi}, \mathbf{w}_1, \ldots, \mathbf{w}_K)$.

Answer: Develop the likelihoood using the IID assumption and using the components t_{nk} in the one-hot encoding of each $\mathbf{t}_n$ as selectors for the right probabilities:

$$p(\mathbf{T}|\mathbf{\Phi}, \mathbf{w}_1, ..., \mathbf{w}_K) \overset{\text{IID}}{=} \prod_{n=1}^{N} p(\mathbf{t}_n|\boldsymbol{\phi}_n, \mathbf{w}_1, ..., \mathbf{w}_K) = \prod_{n=1}^{N}\prod_{k=1}^{K} y_k(\boldsymbol{\phi_n}, \mathbf{w}_1, \ldots, \mathbf{w}_K)^{t_{nk}}.$$

The log-likelihood is then calculate the log-likelihood:

$$\ln p(\mathbf{T}|\mathbf{\Phi}, \mathbf{w}_1, ..., \mathbf{w}_K) = \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_k(\boldsymbol{\phi}_n).$$

(*c*) Derive the gradient of the log-likelihood $\nabla_{\mathbf{w}_j} \log p(\mathbf{T}|\mathbf{\Phi}, \mathbf{w}_1, \ldots, \mathbf{w}_K)$.

Answer: Simirarly to 1a by applying the chain rule and pluggin the results of 1a:

$$\begin{aligned}\frac{\ln p(\mathbf{T}|\mathbf{\Phi}, \mathbf{w}_1, \mathbf{w}_2, .., \mathbf{w}_K)}{\partial \mathbf{w}_j} &= \sum_{n=1}^{N}\sum_{k=1}^{K} \frac{t_{nk}}{y_k}\frac{\partial y_k}{\partial \mathbf{w}_j} \\ &= \sum_{n=1}^{N}\sum_{k=1}^{K} \frac{t_{nk}}{y_k} y_k \boldsymbol{\phi}_n^T(\mathbb{I}[k=j] - y_j) \\ &= \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\boldsymbol{\phi}_n^T(\mathbb{I}[k=j] - y_j) \\ &= \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\boldsymbol{\phi}_n^T\mathbb{I}[k=j] - \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} y_j \boldsymbol{\phi}_n^T \\ &\overset{*}{=} \sum_{n=1}^{N} \boldsymbol{\phi}_n^T t_{nj} - \sum_{n=1}^{N} y_j \boldsymbol{\phi}_n^T \\ &= \sum_{n=1}^{N} (t_{nj} - y_j)\boldsymbol{\phi}_n^T,\end{aligned}$$

where at $\stackrel{*}{=}$ we have $\sum_{k=1}^{K} t_{nk} = 1$, and used that $\mathbb{I}_{kj} = 0$ for $k \neq j$.

(d) What is the objective function we minimize that is equivalent to maximizing the log-likelihood?

Answer: The objective function is the cross-entropy error $E(\mathbf{w}_1, ..., \mathbf{w}_K)$ and is equal to the negative log-likelihood:

$$E(\mathbf{w}_1, ..., \mathbf{w}_K) = -\ln p(\mathbf{T}|\mathbf{\Phi}, \mathbf{w}_1, ..., \mathbf{w}_k) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_k(\boldsymbol{\phi}_n).$$

(e) Write a stochastic gradient algorithm for logistic regression using this objective function. Make sure to include indices for time and to define the learning rate. The gradients may differ in sign switching from maximizing to minimizing; don't overlook this.

Answer: Let us define $E_n(\mathbf{w}_1, \ldots, \mathbf{w}_K) = \sum_{k=1}^{K} t_{nk} \ln y_k(\boldsymbol{\phi}_n)$, and $\nabla_{\mathbf{w}_k} E_n(\mathbf{w}_k) = (t_{nk} - y_k)\boldsymbol{\phi}_n^T$ the gradient to weight vector $\mathbf{w}_k$. The update step for SGD then is:

$$\mathbf{w}_k^{t+1} = \mathbf{w}_k^t - \eta^t (\nabla_{\mathbf{w}_k} E_n(\mathbf{w}_k^t))^T.$$

For T iterations, and by updating the learning rate in each step:

(i) Firstly, initialize $\mathbf{w}_1^0, \mathbf{w}_2^0, ..., \mathbf{w}_k^0$

(ii) Initialize η.

(iii) for t in range (T):

- (A) Randomly choose n from $[1, N]$
- (B) $\mathbf{w}_k^{t+1} = \mathbf{w}_k^t - \eta^t (\nabla_{\mathbf{w}_k} E_n(\mathbf{w}_k^t))^T$ (for all j)
- (C) Decrease η (optional).

(iv) Finally, return $\mathbf{w}^{t=T}$.

(f) In practice, the above vanilla SGD is not effective. Point out a potential weakness of the above algorithm and/or suggest a possible improvement upon it.

Answer:

- The algorithm might converge too slowly if the learning rate is too low.
- When the algorithm becomes close to convergence it will keep oscillating around the minimum due to the stochasticity. To solve this you could include a learning rate scheduling, that decreases the learning rate over time.
- In order to move quicker through valleys of the error functions the algorithm could be enriched with a momentum term.
- The algorithm could be improved by using a different learning rate for each parameter etc ..
