Tarun Krishna
11593040
krishnatarun7@gmail.com

Homework Assignment 3
Machine Learning 1, 17/18

2017-09-27

1 First Exercise

1.1

Calculate $p(\mathbf{T}, \mathbf{X}|\theta)$

$$\begin{aligned} p(\mathbf{T}, \mathbf{X}|\theta) &= \prod_{n=1}^{N} p(t_n, x_n|\theta) \\ &= \prod_{n=1}^{N} p(t_n|\theta) p(x_n|t_n, \theta) \\ &= \prod_{n=1}^{N} \prod_{k=1}^{K} p(C_k|\theta)^{I(t_n=k)} p(x_n|C_k, \theta)^{I(t_n=k)} \end{aligned}$$

more precisely

$$= \prod_{n=1}^{N} \prod_{k=1}^{K} \Big(\prod_{d=1}^{D} p(x_{nd}|C_k, \theta_{dk}) \Big)^{I(t_n=C_k)} p(C_k)^{I(t_n=k)}$$

$p(C_k|\theta_{dk}) = \pi_k$

$$= \prod_{n=1}^{N} \prod_{k=1}^{K} \Big(\prod_{d=1}^{D} p(x_{nd}|C_k, \theta_{dk}) \Big)^{I(t_n=k)} \pi_k^{I(t_n=k)}$$

1.2

Data likelihood $p(\mathbf{T}, \mathbf{X}|\theta)$ for the Poisson model.

$$\begin{aligned} p(\mathbf{T}, \mathbf{X}|\theta) &= \prod_{n=1}^{N} \prod_{k=1}^{K} \Big(\prod_{d=1}^{D} p(x_{nd}|C_k, \theta_{dk}) \Big)^{I(t_n=k)} \pi_k^{I(t_n=k)} \\ &= \prod_{n=1}^{N} \prod_{k=1}^{K} \left(\prod_{d=1}^{D} \frac{\lambda_{dk}^{x_{nd}}}{x_{nd}!} exp(-\lambda_{dk}) \right)^{I(t_n=k)} \pi_k^{I(t_n=k)} \end{aligned}$$

1.3

Log-likelihood for the Poisson model

$$p(\mathbf{T}, \mathbf{X}|\theta) = \prod_{n=1}^{N} \prod_{k=1}^{K} \left(\prod_{d=1}^{D} \frac{\lambda_{dk}^{x_{nd}}}{x_{nd}!} exp(-\lambda_{dk}) \right)^{I(t_n=k)} \pi_k^{I(t_n=k)}$$

Taking log both the sides

$$ln(p(\mathbf{T}, \mathbf{X}|\theta)) = \sum_{n=1}^{N} \sum_{k=1}^{K} \left(I(t_n = k) ln(\pi_k) + I(t_n = k) \sum_{d=1}^{D} \big(-\lambda_{dk} + x_{nd} ln(\lambda_{dk}) - ln(x_{nd}!) \big) \right)$$

1.4

Solve for the MLE estimators for λ_{dk}.

Log-likelihood is given as:

$$= \sum_{n=1}^{N}\sum_{k=1}^{K}\Big(I(t_n = k)ln(\pi_k) + I(t_n = k)\sum_{d=1}^{D}\big(-\lambda_{dk} + x_{nd}ln(\lambda_{dk}) - ln(x_{nd}!)\big)\Big)$$

Takein derivative and equationg to 0

$$0 = \frac{\partial ln(p(\mathbf{T}, \mathbf{X}|\theta))}{\partial \lambda_{dk}}$$

Before we do that log-likelihood can be splitted in to terms depending on λ + independent of λ as:

$$= \sum_{n=1}^{N}\sum_{k=1}^{K} I(t_n = k)ln(\pi_k) + \sum_{n=1}^{N}\sum_{k=1}^{K} I(t_n = k)\sum_{d=1}^{D}\Big(-ln(x_{nd}!) - \lambda_{dk} + x_{nd}ln(\lambda_{dk})\Big)$$

So we can ignore the first term as it is independent of λ

$$\begin{aligned}
0 &= \frac{\partial}{\partial \lambda_{dk}}\sum_{n=1}^{N}\sum_{k=1}^{K} I(t_n = k)\sum_{d=1}^{D}\Big(-ln(x_{nd}!) - \lambda_{dk} + x_{nd}ln(\lambda_{dk})\Big) \\
&= \sum_{n=1}^{N}\Big(-1 + x_{nd}/\lambda_{dk}\Big) \\
\sum_{n=1}^{N} 1 &= 1/\lambda_{dk}\sum_{n=1}^{N} x_{nd} \\
\lambda_{dk} &= \frac{1}{N}\sum_{n=1}^{N} x_{nd}
\end{aligned}$$

1.5

Write $p(C_1|\mathbf{x})$ for the general three class naive Bayes classifier.

$$p(C_1|\mathbf{x}) = \frac{p(\mathbf{x}|C_1)p(C_1)}{\sum_{k=1}^{K=3} p(\mathbf{x}|C_k)p(C_k)}$$

1.6

Write $p(C_1|\mathbf{x})$ for the Poisson model.

$$p(\mathbf{x}|C_k) = \prod_{n=1}^{D}\frac{\lambda_{dk}^{x_d}}{x_d!}exp(-\lambda_{dk})$$

substituting this in $p(C_1|\mathbf{x})$

$$\begin{aligned}
&= \frac{p(C_1)\prod_{n=1}^{D}\frac{\lambda_{d1}^{x_d}}{x_d!}exp(-\lambda_{d1})}{p(C_1)\prod_{n=1}^{D}\frac{\lambda_{d1}^{x_d}}{x_d!}exp(-\lambda_{d1}) + p(C_2)\prod_{n=1}^{D}\frac{\lambda_{d2}^{x_d}}{x_d!}exp(-\lambda_{d2}) + p(C_3)\prod_{n=1}^{D}\frac{\lambda_{d3}^{x_d}}{x_d!}exp(-\lambda_{d3})} \\
&= \frac{\pi_1\prod_{n=1}^{D}\frac{\lambda_{d1}^{x_d}}{x_d!}exp(-\lambda_{d1})}{\pi_1\prod_{n=1}^{D}\frac{\lambda_{d1}^{x_d}}{x_d!}exp(-\lambda_{d1}) + \pi_2\prod_{n=1}^{D}\frac{\lambda_{d2}^{x_d}}{x_d!}exp(-\lambda_{d2}) + \pi_3\prod_{n=1}^{D}\frac{\lambda_{d3}^{x_d}}{x_d!}exp(-\lambda_{d3})}
\end{aligned}$$

1.7

$$p(C_1|\mathbf{x}) > p(C_2|\mathbf{x})$$
$$p(C_1|\mathbf{x}) > p(C_3|\mathbf{x})$$

Will prove one and second can be symmetrically derived

$$\frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x})} > \frac{p(\mathbf{x}|C_2)p(C_2)}{p(\mathbf{x})}$$

Considering Poisson model

$$p(C_1)\prod_{d=1}^{D}\frac{\lambda_{d1}^{x_d}}{x_d!}exp(-\lambda_{d1}) > p(C_2)\prod_{d=1}^{D}\frac{\lambda_{d2}^{x_d}}{x_d!}exp(-\lambda_{d2})$$

Taking constants on one side

$$\prod_{d=1}^{D}(\frac{\lambda_{d1}}{\lambda_{d2}})^{x_d}\exp(\lambda_{d2}-\lambda_{d1}) > constant$$

taking log both sides

$$\sum_{n=1}^{D}x_d ln(\frac{\lambda_{d1}}{\lambda_{d2}}) + \sum_{d}^{D}(\lambda_{d2}-\lambda_{d1}) > constant$$

similarly for other inequaltiy we get

$$\sum_{n=1}^{D}x_d ln(\frac{\lambda_{d1}}{\lambda_{d3}}) + \sum_{d}^{D}(\lambda_{d3}-\lambda_{d1}) > constant$$
$$\mathbf{x}^T a > c$$

Where **x** is a vector with d elements **a** is a vector with each elements log of ratios.Same result can be derived for the other inequality as well

1.8

We already used an inequaltiy if **x** is in C_1 which is :

$$p(C_1|\mathbf{x}) > p(C_2|\mathbf{x})$$
$$p(C_1|\mathbf{x}) > p(C_3|\mathbf{x})$$

Which states that if $\mathbf{x}_a$ and $\mathbf{x}_b$ are lying in C_1, then any point $\mathbf{x}_c$ on the line joining $\mathbf{x}_a$,and $\mathbf{x}_b$ will has to be in region C_1. Because both $\mathbf{x}_a$ and $\mathbf{x}_b$ lie inside region C_1, it follows that $y_1(\mathbf{x}_a) > y_j(\mathbf{x}_a)$ and $y_1(\mathbf{x}_b) > y_j(\mathbf{x}_b)$ for all $j \neq 1$,so $\mathbf{x}_c$ also lies in region C_1 .That our region is Convex

1.9

Let us consider medical diagnosis problem. We note that, if a patient who does not have cancer is incorrectly diagnosed as having cancer, the consequences may be some patient distress plus the need for further investigations. Conversely, if a patient with cancer is diagnosed as healthy, the result may be premature death due to lack of treatment. Thus the consequences of these two types of mistake can be dramatically different. It would clearly be better to make fewer mistakes of the second kind, even if this was at the expense of making more mistakes of the first kind. So in the cases where it is very likely for machine to go wrong or their is some kind of ambiguity as mentioned above it is better to have a human internvention.

2 Second Exercise

2.1 First Subtask

$$
\begin{aligned}
\frac{\partial y_k}{\partial \mathbf{a}_j} &= y_k(I_{kj} - y_j) \textbf{Bishop 4.106} \\
a_j &= \mathbf{w}_j^T \phi \\
\frac{\partial y_k}{\partial \mathbf{w}_j} &= \frac{\partial y_k}{\partial a_j}\frac{\partial a_j}{\partial \mathbf{w}_j} \\
\frac{\partial y_k}{\partial \mathbf{w}_j} &= y_k(I_{kj} - y_j)\frac{\partial a_j}{\partial \mathbf{w}_j} \\
&= y_k(I_{kj} - y_j)\phi^T
\end{aligned}
$$

2.2

Likelihood :

$$
\begin{aligned}
p(\mathbf{T}|\mathbf{w}_1, ..., \mathbf{w}_K) &= \prod_{n=1}^{N}\prod_{k=1}^{K}(p(C_k|\phi_n))^{t_{nk}} \\
ln(p(\mathbf{T}|\mathbf{w}_1, ..., \mathbf{w}_K)) &= \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} ln(p(C_k|\phi_n)) \\
ln(p(\mathbf{T}|\mathbf{w}_1, ..., \mathbf{w}_K)) &= \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} ln(y_{nk})
\end{aligned}
$$

where $y_{nk} = y_k(\phi_n)$

2.3

Gradient of the negative log-likelihood. Will be using two results already derived i.e:

$$
\begin{aligned}
\frac{\partial y_k}{\partial a_j} &= y_k(I_{kj} - y_j) \\
\frac{\partial y_k}{\partial \mathbf{w}_j} &= y_k(I_{kj} - y_j)\phi^T
\end{aligned}
$$

Given above results we can proceed further also $\mathbf{w}_i$ is a vector(i in general)

$$
\begin{aligned}
\frac{\partial \mathbf{E}}{\partial \mathbf{w}_j} &= -\frac{\partial}{\partial \mathbf{w}_j}\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} ln(y_{nk}) \\
&= -\sum_{n=1}^{N}\sum_{k=1}^{K}\frac{t_{nk}}{y_{nk}}(y_{nk}(I_{kj} - y_{nj}))\phi^T \\
&= -\sum_{n=1}^{N}\Bigg(\underbrace{\sum_{k=1}^{K} t_{nk} I_{kj}}_{t_{nj}} - \sum_{k=1}^{K} t_{nk} y_{nj}\Bigg)\phi^T \\
&= -\sum_{n=1}^{N}\big(t_{nj} - y_{nj}\big)\phi^T
\end{aligned}
$$

taking the negative sign in

$$= \sum_{n=1}^{N} (y_{nj} - t_{nj})\phi^T$$

$$\nabla_{\mathbf{w}_j} \mathbf{E} = \left(\frac{\partial \mathbf{E}}{\partial \mathbf{w}_j} \right)^T$$

$$\nabla_{\mathbf{w}_j} \mathbf{E} = \sum_{n=1}^{N} (y_{nj} - t_{nj})\phi$$

2.4

Function we minimize that is equivalent to maximizing the log likelihood.

$$E(\mathbf{w}_1, ..., \mathbf{w}_K) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} ln(y_{nk})$$

2.5

Stochastic gradient algorithm for logistic regression using this objective function.

η : learning rate

for all j :

$$\mathbf{w}_j^{\tau} = \mathbf{w}_j^{\tau - 1} - \eta \nabla_{\mathbf{w}_j} \mathbf{E}$$

$j \in \{1, 2..K\}$

Well this update is done for every $\mathbf{w}_k$ for each iteration. With each iteration I mean no of times you want to run gradient update $\mathbf{w}_1$,$\mathbf{w}_2$...$\mathbf{w}_k$ depending upon the convergence. You can run it for 1000 iteration or even depending upon convergence etc.

2.6

In gradient descent we start at some point on the error function defined over the weights, and attempt to move to the global minimum of the function. Any step in a downward direction will take us closer to the global minimum. For real problems, however, error surfaces are typically complex. Here there are numerous local minima. Progress here is only possible by climbing higher before descending to the global minimum. Precisely Plain SGD can make erratic updates on non-smooth loss functions. One way is to add momentum [1] .This is probably the most popular extension of the backprop algorithm; it is hard to find cases where this is not used.

[1] On the importance of initialization and momentum in deep learning