Andrii Skliar
11636785
andrii.skliar@student.uva.nl

Homework Assignment 1

Machine Learning 1, 17/18

2019-01-08

1 Naive Bayes Spam Classification

1.

$$\begin{aligned}
p(\mathbf{T},\mathbf{X}|\mathbf{\Theta}) &= p(\mathbf{X}|\mathbf{T},\mathbf{\Theta})p(\mathbf{T}) \\
&= \prod_{n=1}^{N}\prod_{k=1}^{3}\left(\prod_{d=1}^{D} p(x_{nd}|C_k,\theta_{dk})\right)^{\mathbb{1}(t_n=k)} \cdot \prod_{n=1}^{N}\prod_{k=1}^{3} p(C_k)^{\mathbb{1}(t_n=k)} \\
&= \prod_{n=1}^{N}\prod_{k=1}^{3}\left(\left(\prod_{d=1}^{D} p(x_{nd}|C_k,\theta_{dk})\right)p(C_k)\right)^{\mathbb{1}(t_n=k)} \\
&= \prod_{n=1}^{N}\prod_{k=1}^{3}\left(\left(\prod_{d=1}^{D} p(x_{nd}|C_k,\theta_{dk})\right)\pi_k\right)^{\mathbb{1}(t_n=k)}
\end{aligned}$$

2.

$$\begin{aligned}
p(\mathbf{T},\mathbf{X}|\mathbf{\Theta}) &= \prod_{n=1}^{N}\prod_{k=1}^{3}\left(\left(\prod_{d=1}^{D} p(x_{nd}|C_k,\theta_{dk})\right)\pi_k\right)^{\mathbb{1}(t_n=k)} \\
&= \prod_{n=1}^{N}\prod_{k=1}^{3}\left(\left(\prod_{d=1}^{D} \frac{\lambda_{dk}^{x_{nd}}}{x_{nd}!} exp(-\lambda_{dk})\right)\pi_k\right)^{\mathbb{1}(t_n=k)}
\end{aligned}$$

3.

$$\log p(\mathbf{T},\mathbf{X}|\mathbf{\Theta}) = \sum_{n=1}^{N}\sum_{k=1}^{3}\left(\sum_{d=1}^{D}(x_{nd}\log\lambda_{dk} - \log x_{nd}! - \lambda_{dk}) + \log\pi_k\right)\mathbb{1}(t_n=k)$$

4.

$$\begin{aligned}
\lambda_{dk_{ML}} &= \arg\max_{\lambda_{dk}} p(\mathbf{T},\mathbf{X}|\mathbf{\Theta}) \\
&= \arg\max_{\lambda_{dk}} log\left((p(\mathbf{T},\mathbf{X}|\mathbf{\Theta})\right) \\
\frac{\partial \log p(\mathbf{T},\mathbf{X}|\mathbf{\Theta})}{\partial\lambda_{dk}} &= (\text{because we differentiate by } \lambda_{dk}, \\
&\text{both d and k will be fixed and thus only sum over N will be left}) \\
&= \sum_{n=1}^{N}\left(\frac{x_{nd}}{\lambda_{dk}} - 1\right)\mathbb{1}(t_n=k) \\
&= \sum_{n=1}^{N}\left(\mathbb{1}(t_n=k)\frac{x_nd}{\lambda_{dk}} - \mathbb{1}(t_n=k)\right) \\
&= \frac{\sum_{n=1}^{N}\mathbb{1}(t_n=k)x_{nd}}{\lambda_{dk}} - \sum_{n=1}^{N}\mathbb{1}(t_n=k) = 0 \\
\sum_{n=1}^{N}\mathbb{1}(t_n=k)x_{nd} &= \lambda_{dk_{ML}}\sum_{n=1}^{N}\mathbb{1}(t_n=k)
\end{aligned}$$

If $\sum_{n=1}^{N} \mathbb{1}(t_n = k) \neq 0$, we can divide both parts by it and we will have following expression for MLE then:

$$\lambda_{dk_{ML}} = \frac{\sum_{n=1}^{N} \mathbb{1}(t_n = k) x_{nd}}{\sum_{n=1}^{N} \mathbb{1}(t_n = k)}$$

5.

$$\begin{aligned}
p(C_1|\mathbf{x}) &= \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x})} \\
&= \frac{p(\mathbf{x}|C_1)p(C_1)}{\sum_{k=1}^{3} p(x|C_k)p(C_k)} \\
&= \frac{\prod_{d=1}^{D} \left(p(x_d|C_1)\right) p(C_1)}{\sum_{k=1}^{3} \left(\prod_{d=1}^{D} \left(p(x_d|C_k)\right) p(C_k)\right)} \\
&= \frac{\prod_{d=1}^{D} \left(p(x_d|C_1)\right) \pi_1}{\sum_{k=1}^{3} \left(\prod_{d=1}^{D} \left(p(x_d|C_k)\right) \pi_k\right)}
\end{aligned}$$

6.

$$\begin{aligned}
p(C_1|\mathbf{x}, \mathbf{\Theta}) &= \frac{p(\mathbf{x}|C_1, \mathbf{\Theta})p(C_1)}{\sum_{k=1}^{3} p(x|C_k, \mathbf{\Theta})p(C_k)} \\
&= \frac{\prod_{d=1}^{D} \left(\frac{\lambda_{d1}^{x_d}}{x_d!} e^{-\lambda_{d1}}\right) \pi_1}{\sum_{k=1}^{3} \left(\prod_{d=1}^{D} \left(\frac{\lambda_{dk}^{x_d}}{x_d!} e^{-\lambda_{dk}}\right) \pi_k\right)}
\end{aligned}$$

7.

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

$$\frac{\prod_{d=1}^{D} \left(p(x_d|C_1, \theta_{d1})\right) \pi_1}{\sum_{k=1}^{3} \left(\prod_{d=1}^{D} \left(p(x_d|C_k, \theta_{dk})\right) \pi_k\right)} > \frac{\prod_{d=1}^{D} \left(p(x_d|C_2, \theta_{d2})\right) \pi_2}{\sum_{k=1}^{3} \left(\prod_{d=1}^{D} \left(p(x_d|C_k, \theta_{dk})\right) \pi_k\right)}$$

$$\prod_{d=1}^{D} \left(p(x_d|C_1, \theta_{d1})\right) \pi_1 > \prod_{d=1}^{D} \left(p(x_d|C_2, \theta_{d2})\right) \pi_2$$

$$\prod_{d=1}^{D} \left(\frac{\lambda_{d1}^{x_d}}{x_d!} e^{-\lambda_{d1}}\right) \pi_1 > \prod_{d=1}^{D} \left(\frac{\lambda_{d2}^{x_d}}{x_d!} e^{-\lambda_{d2}}\right) \pi_1$$

$$\log \prod_{d=1}^{D} \left(\frac{\lambda_{d1}^{x_d}}{x_d!} e^{-\lambda_{d1}}\right) \pi_1 > \log \prod_{d=1}^{D} \left(\frac{\lambda_{d2}^{x_d}}{x_d!} e^{-\lambda_{d2}}\right) \pi_1$$

$$\left(\sum_{d=1}^{D} (x_d log \lambda_{d1} - log x_d! - \lambda_{d1})\right) + \log \pi_1 > \left(\sum_{d=1}^{D} (x_d log \lambda_{d2} - log x_d! - \lambda_{d2})\right) + \log \pi_2$$

$$\sum_{d=1}^{D} \left(x_d(log\lambda_{d1} - log\lambda_{d2}) + (-\lambda_{d1} + \lambda_{d2})\right) + log\pi_1 - log\pi_2 = \sum_{d=1}^{D} \left(x_d log\frac{\lambda_{d1}}{\lambda_{d2}} - \lambda_{d1} + \lambda_{d2}\right) + log\frac{\pi_1}{\pi_2} > 0$$

$$\sum_{d=1}^{D} x_d log\frac{\lambda_{d1}}{\lambda_{d2}} > \sum_{d=1}^{D} (\lambda_{d1} - \lambda_{d2}) - log\frac{\pi_1}{\pi_2}$$

$$\sum_{d=1}^{D} x_d log\frac{\lambda_{d1}}{\lambda_{d2}} > \sum_{d=1}^{D} (\lambda_{d1} - \lambda_{d2}) + log\frac{\pi_2}{\pi_1}$$

$$\begin{aligned}
&\mathbf{x}^T\mathbf{a_1} > c_1,\\
where\ c_1 &= \sum_{d=1}^{D}(\lambda_{d1}-\lambda_{d2}) + log\frac{\pi_2}{\pi_1};\\
\mathbf{x} &= (x_1, x_2, \dots x_D)^T;\\
\mathbf{a_1} &= \left(log\frac{\lambda_{11}}{\lambda_{12}}, log\frac{\lambda_{21}}{\lambda_{22}}, \dots log\frac{\lambda_{D1}}{\lambda_{D2}}\right)^T
\end{aligned}$$

The same approach will also work for $p(C_1|x) > p(C_3|x)$ and therefore we can write $\mathbf{x}^T\mathbf{a} > c$ in a pretty straightforward manner:

$$\begin{aligned}
&\mathbf{x}^T\mathbf{a_2} > c_2,\\
where\ c_2 &= \sum_{d=1}^{D}(\lambda_{d1}-\lambda_{d3}) + log\frac{\pi_3}{\pi_1};\\
\mathbf{x} &= (x_1, x_2, \dots x_D)^T;\\
\mathbf{a_2} &= \left(log\frac{\lambda_{11}}{\lambda_{13}}, log\frac{\lambda_{21}}{\lambda_{23}}, \dots log\frac{\lambda_{D1}}{\lambda_{D3}}\right)^T
\end{aligned}$$

So, $\mathbf{x}$ belongs to class C_1 if following constraints are satisfied:

$$\begin{cases}\mathbf{x}^T\mathbf{a_1} > c_1\\ \mathbf{x}^T\mathbf{a_2} > c_2\end{cases}$$

8.

Yes, region where $\mathbf{x}$ is predicted to be in C_1 is convex.
To prove this, we will use standard procedure. It will be shown for $\mathbf{x}^T\mathbf{a_1} > c_1$ and $\mathbf{x}^T\mathbf{a_2} > c_2$.
Note: for the purpose of convenience, we will call region where $\mathbf{x}$ is predicted to be in C_1 as R_1

- $\mathbf{x}_A, \mathbf{x}_B \in R_1$
- $\hat{\mathbf{x}} = \lambda\mathbf{x}_A + (1-\lambda)\mathbf{x}_B, 0 \le \lambda \le 1$
- R_1 is convex if $\hat{\mathbf{x}}$ also lies in R_1.

For $\hat{\mathbf{x}}^T\mathbf{a_1} > c_1$:

$$\begin{aligned}
\hat{\mathbf{x}}^T\mathbf{a_1} &= (\lambda\mathbf{x}_A + (1-\lambda)\mathbf{x}_B)\mathbf{a_1}\\
&= \lambda\mathbf{x}_A^T\mathbf{a_1} + (1-\lambda)\mathbf{x}_B^T\mathbf{a_1}\\
c_1 &= (\lambda + 1 - \lambda)c_1\\
&= \lambda c_1 + (1-\lambda)c_1
\end{aligned}$$

Because $\mathbf{x}_A, \mathbf{x}_B \in R_1$:

$$\begin{cases}\mathbf{x}_A^T\mathbf{a_1} > c_1\\ \mathbf{x}_B^T\mathbf{a_1} > c_1\end{cases} \quad \begin{cases}\lambda\mathbf{x}_A^T\mathbf{a_1} > \lambda c_1\\ (1-\lambda)\mathbf{x}_B^T\mathbf{a_1} > (1-\lambda)c_1\end{cases} \quad \Rightarrow \hat{\mathbf{x}}^T\mathbf{a_1} > c_1$$

For $\hat{\mathbf{x}}^T\mathbf{a_2} > c_2$:

$$\begin{aligned}
\hat{\mathbf{x}}^T\mathbf{a_2} &= (\lambda\mathbf{x}_A + (1-\lambda)\mathbf{x}_B)\mathbf{a_2}\\
&= \lambda\mathbf{x}_A^T\mathbf{a_2} + (1-\lambda)\mathbf{x}_B^T\mathbf{a_2}\\
c_2 &= (\lambda + 1 - \lambda)c_2\\
&= \lambda c_2 + (1-\lambda)c_2
\end{aligned}$$

Because $\mathbf{x}_A, \mathbf{x}_B \in R_1$:

$$\begin{cases}\mathbf{x}_A^T\mathbf{a_2} > c_2\\ \mathbf{x}_B^T\mathbf{a_2} > c_2\end{cases} \quad \begin{cases}\lambda\mathbf{x}_A^T\mathbf{a_2} > \lambda c_2\\ (1-\lambda)\mathbf{x}_B^T\mathbf{a_2} > (1-\lambda)c_2\end{cases} \quad \Rightarrow \hat{\mathbf{x}}^T\mathbf{a_2} > c_2$$

Because both constraints $\hat{\mathbf{x}}^T\mathbf{a_1} > c_1$ and $\hat{\mathbf{x}}^T\mathbf{a_2} > c_2$ are satisfied, $\hat{\mathbf{x}}$ lies in R_1 and therefore, R_1 is convex.

9.

There are many examples of the applications where you would use human' help for ambiguous predictions, but the first one that comes up to my mind and that seems the most logical for me is medical sphere, because the price of the mistake there is too high. For example, let's say we have an application which classifies lungs tumor as malignant or benign using x-ray images. If at some point the classifier is not sure whether the tumor presented is malignant or benign, it is reasonable to ask doctors for help other than just choosing the result randomly.
Another application is the training of the pre-trained algorithms. What I mean is following: let's say we have some website, for example, Quora. The purpose of our algorithm is to make topic classification for the questions. We have already trained it on some labeled data and it is working properly. However, at some point, it encounters the question, that it can't classify and then it would be useful for a human to manually label it so that serves as an additional training data for our current classifier.

2 Multi-class Logistic Regression

Note:

- $y_k(\phi) = p(C_k|\boldsymbol{\phi}) = \frac{exp(a_k)}{\sum_i exp(a_i)}$
- K - the number of classes
- $\boldsymbol{\phi} = \boldsymbol{\phi}(\mathbf{x}) = (\phi_0(\mathbf{x}), \phi_1(\mathbf{x}), \dots \phi_{M-1}(\mathbf{x}))^T$
- $\boldsymbol{\phi}_n(\mathbf{x}) = (\phi_0(\mathbf{x}_n), \phi_1(\mathbf{x}_n), \dots \phi_{M-1}(\mathbf{x}_n))^T$
- $a_k = \mathbf{w}_k^T \boldsymbol{\phi}$
- $\mathbf{T} = (\mathbf{t}_1, \mathbf{t}_2, \dots \mathbf{t}_N)^T$
- $\boldsymbol{\Phi} = (\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \dots \boldsymbol{\phi}_N)^T$

1.

$$\begin{aligned}\frac{\partial y_k}{\partial \mathbf{w_j}} &= \frac{\boldsymbol{\phi}^T I_{kj} exp(a_k) \sum_i exp(a_i) - \boldsymbol{\phi}^T exp(a_k) exp(a_j)}{\left(\sum_i exp(a_i)\right)^2} \\ &= \frac{exp(a_k)\boldsymbol{\phi}^T}{\sum_i exp(a_i)} \left(I_{kj} - \frac{exp(a_j)}{\sum_i exp(a_i)} \right) \\ &= y_k(\boldsymbol{\phi})(I_{kj} - y_j(\boldsymbol{\phi}))\boldsymbol{\phi}^T\end{aligned}$$

2.

$$p(\mathbf{T}|\boldsymbol{\Phi}, \mathbf{w_1}, \mathbf{w_2}, \dots \mathbf{w_K}) = \prod_{n=1}^{N} \prod_{k=1}^{K} p(C_k|\boldsymbol{\phi}_n)^{t_{nk}} = \prod_{n=1}^{N} \prod_{k=1}^{K} y_{nk}^{t_{nk}}$$

$$\log p(\mathbf{T}|\boldsymbol{\Phi}, \mathbf{w_1}, \mathbf{w_2}, \dots \mathbf{w_K}) = \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \log y_{nk}$$

3.

$$
\begin{aligned}
\nabla_{w_j} \log p(\mathbf{T}|\mathbf{\Phi}, \mathbf{w_1}, \mathbf{w_2}, \dots \mathbf{w_k}) &= \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \frac{\cancel{y_{nk}}(I_{kj} - y_{nj})\boldsymbol{\phi}_n}{\cancel{y_{nk}}} \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} I_{kj} \boldsymbol{\phi}_n - \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} y_{nj} \boldsymbol{\phi}_n \\
&= \sum_{n=1}^{N} t_{nj} \boldsymbol{\phi}_n - \sum_{n=1}^{N} y_{nj} \boldsymbol{\phi_n} \sum_{k=1}^{K} t_{nk} \\
&= \left(\text{because } \sum_{k=1}^{K} t_{nk} = 1, \text{ we can simplify our expression} \right) \\
&= \sum_{n=1}^{N} t_{nj} \boldsymbol{\phi}_n - \sum_{n=1}^{N} y_{nj} \boldsymbol{\phi_n} \\
&= \sum_{n=1}^{N} (t_{nj} - y_{nj}) \boldsymbol{\phi}_n
\end{aligned}
$$

4.

We are minimizing the cross entropy function $E(\mathbf{w_1}, \mathbf{w_2}, \dots \mathbf{w_K})$, which is equal to the negative log-likelihood. The function has the following form:

$$E(\mathbf{w_1}, \mathbf{w_2}, \dots \mathbf{w_K}) = -\log p(\mathbf{T}|\mathbf{\Phi}, \mathbf{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \log y_{nk}$$

If we take the gradient by w_j, we obtain the following function:

$$\nabla_{w_j} E(\mathbf{w_1}, \mathbf{w_2}, \dots \mathbf{w_K}) = -\nabla_{w_j} \log p(\mathbf{T}|\mathbf{\Phi}, \mathbf{w}) = \sum_{n=1}^{N} (y_{nj} - t_{nj}) \boldsymbol{\phi}_n$$

5.

Note:

- $\mathbf{W} = (\mathbf{w_1}, \mathbf{w_2}, \dots \mathbf{w_K})$.
- K is the number of possible classes.

Algorithm 1 Multiclass Stochastic Gradient Descent

```
 1: Initialize $\mathbf{W}^{(0)}$
 2: Set time: $\tau = 0$
 3: Choose a learning rate $\eta$
 4: while $||\mathbf{W}^{\tau+1} - \mathbf{W}^{\tau}|| > \epsilon$ do
 5:     Choose a random data point $(\mathbf{x}_n, t_n)$
 6:     for j := 1 to $K$ do
 7:         Update $\mathbf{w}_j$: $\mathbf{w}_j^{(\tau+1)} = \mathbf{w}_j^{(\tau)} - \eta(y_{nj} - t_{nj})\phi(\mathbf{x}_n)$
 8:     end for
 9:     Update time: $\tau = \tau + 1$
10: end while
11: return $\mathbf{W}^{(\tau)}$
```

6.

The potential drawbacks are following:

- SGD might get stuck in a local minimum of a function and it can therefore be problematic to find solution for non-convex problems. The solution to that might be introducing momentum and/or random re-starts to skip local minimums and find the global one.

- SGD is greatly dependent on the learning rate. If it is too small, it might take a long time to converge to the local minimum. If it is too big, the algorithm might skip the true local minimum and not converge at all. The possible solution for that might be introducing learning rate decay, meaning, that learning rate will get smaller when the error gets smaller.
- SGD depends on the order in which we present the points to it. Therefore, depending on the points, it can take different time for the algorithm to converge. That problem might be solved using batch gradient descent, where you present not one point at a time to the algorithm, but more than one (i.e 10 points in one iteration).