Yke Rusticus
11306386
yke.rusticus@student.uva.nl

Homework Assignment 3
Machine Learning 1, 19/20

2019-10-03

1 Naive Bayes Text Classification

1.1

$\mathbf{T} \in \mathbb{R}^{N\times K}$, $\mathbf{X} \in \mathbb{R}^{N\times D}$, $\boldsymbol{\theta} \in \mathbb{R}^{D\times K}$.

$$\begin{aligned}
p(\mathbf{T},\mathbf{X} \mid \boldsymbol{\theta}) &= \prod_{n=1}^{N} p(\mathbf{t}_n, \mathbf{x}_n \mid \boldsymbol{\theta}) && (1)\\
&= \prod_{n=1}^{N} p(\mathbf{t}_n \mid \boldsymbol{\theta}) p(\mathbf{x}_n \mid \mathbf{t}_n, \boldsymbol{\theta}) \quad \text{product rule} && (2)\\
&= \prod_{n=1}^{N}\prod_{k=1}^{K} \Big[p(t_{nk} \mid \boldsymbol{\theta}) p(\mathbf{x}_n \mid t_{nk}, \boldsymbol{\theta})\Big]^{t_{nk}} && (3)\\
&= \prod_{n=1}^{N}\prod_{k=1}^{K} \Big[\pi_k \prod_{d=1}^{D} p(x_{nd} \mid t_{nk}, \theta_{dk})\Big]^{t_{nk}} \quad \text{NB assumption} && (4)\\
&= \prod_{n=1}^{N}\prod_{k=1}^{K} \Big[\pi_k \prod_{d=1}^{D} p(x_{nd} \mid \mathcal{C}_k, \theta_{dk})\Big]^{t_{nk}} && (5)
\end{aligned}$$

1.2

Without the naive Bayes assumption, we would have for each class k a number of D parameters plus all their correlations, resulting in a total of $K(2^D - 1)$ parameters. Leaving out all those correlations, i.e. using the naive Bayes assumption, results then in a total of KD parameters. The reason why this is called "naive" is because often there is no such independence in the data that is given. For example in document classification, almost all words have some correlation. As we can see from this (previous) sentence, the word "correlation" has a strong correlation with the rest of the sentence, and it makes a weird sentence if one assumes no correlation and fills in a random word like "staircase" instead.

1.3

Likelihood:

$$p(\mathbf{T},\mathbf{X} \mid \boldsymbol{\theta}) = \prod_{n=1}^{N}\prod_{k=1}^{K} \Big[\pi_k \prod_{d=1}^{D} \theta_{dk}^{x_{nd}} (1-\theta_{dk})^{1-x_{nd}}\Big]^{t_{nk}} \tag{6}$$

Log-likelihood:

$$\begin{aligned}
\ln p(\mathbf{T},\mathbf{X} \mid \boldsymbol{\theta}) &= \sum_{n=1}^{N}\sum_{k=1}^{K} \ln \Big[\pi_k \prod_{d=1}^{D} \theta_{dk}^{x_{nd}} (1-\theta_{dk})^{1-x_{nd}}\Big]^{t_{nk}} && (7)\\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\Big[\ln \pi_k + \sum_{d=1}^{D} \ln\big(\theta_{dk}^{x_{nd}} (1-\theta_{dk})^{1-x_{nd}}\big)\Big] && (8)\\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\Big[\ln \pi_k + \sum_{d=1}^{D} \big[x_{nd}\ln\theta_{dk} + (1-x_{nd})\ln(1-\theta_{dk})\big]\Big] && (9)
\end{aligned}$$

1.4

$$\frac{\partial \ln p(\mathbf{T},\mathbf{X} \mid \boldsymbol{\theta})}{\partial \theta_{dk}} = \frac{\partial}{\partial \theta_{dk}} \sum_{n=1}^{N}\sum_{k'=1}^{K} t_{nk'}\Big[\ln \pi_{k'} + \sum_{d'=1}^{D} \big[x_{nd'}\ln\theta_{d'k'} + (1-x_{nd'})\ln(1-\theta_{d'k'})\big]\Big] \tag{10}$$

From this we can see that only the terms where $k' = k$ and $d' = d$ will contribute to the derivative. Also t_{nk} will be zero for each n that does not belong to class k, so we can write:

$$
\begin{aligned}
\frac{\partial}{\partial\theta_{dk}} \sum_{n\in\mathcal{C}_k}^{N} \left[\ln \pi_k + x_{nd}\ln\theta_{dk} + (1-x_{nd})\ln(1-\theta_{dk})\right] && (11)\\
= \sum_{n\in\mathcal{C}_k}^{N} \left[\frac{x_{nd}}{\theta_{dk}} - \frac{1-x_{nd}}{1-\theta_{dk}}\right] \overset{!}{=} 0 && (12)\\
\sum_{n\in\mathcal{C}_k}^{N} \frac{x_{nd}}{\theta_{dk}} = \sum_{n\in\mathcal{C}_k}^{N} \frac{1-x_{nd}}{1-\theta_{dk}} && (13)\\
(1-\theta_{dk}) \sum_{n\in\mathcal{C}_k}^{N} x_{nd} = \theta_{dk} \sum_{n\in\mathcal{C}_k}^{N} (1-x_{nd}) && (14)\\
\frac{1}{\theta_{dk}} \sum_{n\in\mathcal{C}_k}^{N} x_{nd} - \sum_{n\in\mathcal{C}_k}^{N} x_{nd} = \sum_{n\in\mathcal{C}_k}^{N} (1-x_{nd}) && (15)\\
\frac{1}{\theta_{dk}} \sum_{n\in\mathcal{C}_k}^{N} x_{nd} = \sum_{n\in\mathcal{C}_k}^{N} 1 := N_k && (16)\\
\theta_{dk} = \frac{1}{N_k} \sum_{n\in\mathcal{C}_k}^{N} x_{nd}, && (17)
\end{aligned}
$$

which is the MLE estimator for θ_{dk}.
Interpretation: x_{nd} is 1 if word d is present in document n, and 0 if not, so $\sum_{n\in\mathcal{C}_k}^{N} x_{nd}$ counts the number of times word d is present in all the documents that belong to class k. N_k is the number of documents of class k. Therefore, θ_{dk} represents the average frequency of word d in documents of class k.

1.5

$$
\begin{aligned}
p(\mathcal{C}_1 \mid \mathbf{x}) &= \frac{p(\mathbf{x} \mid \mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x})} && (18)\\
p(\mathcal{C}_1) &= \pi_1 && (19)\\
p(\mathbf{x} \mid \mathcal{C}_1) &= \prod_{d=1}^{D} p(x_d \mid \mathcal{C}_1) && (20)\\
p(\mathbf{x}) &= \sum_{k=1}^{K} p(\mathbf{x} \mid \mathcal{C}_k)p(\mathcal{C}_k) && (21)\\
p(\mathcal{C}_1 \mid \mathbf{x}) &= \frac{\pi_1 \prod_{d=1}^{D} p(x_d \mid \mathcal{C}_1)}{\sum_{k=1}^{K} p(\mathbf{x} \mid \mathcal{C}_k)p(\mathcal{C}_k)} && (22)
\end{aligned}
$$

1.6

$$
p(\mathcal{C}_1 \mid \mathbf{x}) = \frac{\pi_1 \prod_{d=1}^{D} \theta_{d1}^{x_d}(1-\theta_{d1})^{1-x_d}}{\sum_{k=1}^{K} \pi_k \prod_{d=1}^{D} \theta_{dk}^{x_d}(1-\theta_{dk})^{1-x_d}} \tag{23}
$$

2 Multi-class Logistic Regression and Multilayer Perceptrons

2.1

Likelihood [Bishop p.209]:

$$p(\mathbf{T} \mid \mathbf{\Phi}, \mathbf{w}_1, \dots, \mathbf{w}_K) = \prod_{n=1}^{N} \prod_{k=1}^{K} y_k(\phi_n)^{t_{nk}}, \tag{24}$$

where $y_k(\phi_n) = \exp a_k / \sum_j \exp a_j$, and $a_i = \mathbf{w}_i^\top \phi_n$.
Log-likelihood:

$$\begin{aligned} \ln \prod_{n=1}^{N} \prod_{k=1}^{K} y_k(\phi_n)^{t_{nk}} &= \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln y_k(\phi_n) & (25)\\ &= \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} (a_k - \ln \sum_j \exp a_j) & (26) \end{aligned}$$

Derivative of y_k w.r.t. $\mathbf{w}_j$:

$$\begin{aligned} \frac{\partial y_k}{\partial \mathbf{w}_j} &= \frac{\partial y_k}{\partial a_j} \frac{\partial a_j}{\partial \mathbf{w}_j} \quad \text{since} \quad \frac{\partial a_i}{\partial \mathbf{w}_j} = \mathbf{0}^\top & (27)\\ \frac{\partial y_k}{\partial a_j} &= \frac{\delta_{kj} \exp a_k \sum_i \exp a_i - \exp a_k \exp a_j}{(\sum_i \exp a_i)^2} & (28)\\ &= \frac{\delta_{kj} \exp a_k}{\sum_i \exp a_i} - \frac{\exp a_k \exp a_j}{\sum_i \exp a_i} & (29)\\ &= y_k(\delta_{kj} - y_j) & (30)\\ \frac{\partial a_j}{\partial \mathbf{w}_j} &= \frac{\partial}{\partial \mathbf{w}_j} \mathbf{w}_j^\top \phi_n = \phi_n^\top & (31)\\ \frac{\partial y_k}{\partial \mathbf{w}_j} &= y_k(\delta_{kj} - y_j)\phi_n^\top & (32) \end{aligned}$$

Gradient of the log-likelihood w.r.t. $\mathbf{w}_j$:

$$\begin{aligned} \nabla_{\mathbf{w}_j} \ln p(\mathbf{T} \mid \mathbf{\Phi}, \mathbf{w}_1, \dots, \mathbf{w}_K) &= \sum_{n=1}^{N} \sum_{k=1}^{K} \frac{t_{nk}}{y_k} \frac{\partial y_k}{\partial \mathbf{w}_j} \quad \text{chain rule} & (33)\\ &= \sum_{n=1}^{N} \sum_{k=1}^{K} \frac{t_{nk}}{y_k} y_k(\delta_{kj} - y_j)\phi_n^\top & (34)\\ &= \sum_{n=1}^{N} (t_{nj} - y_j)\phi_n^\top & (35)\\ &= (\mathbf{T}_{:,j} - \mathbf{Y}_{:,j})^\top \mathbf{\Phi} \in \mathbb{R}^{1xM}, & (36) \end{aligned}$$

such that $y_j(\phi_n)$ is equal to element Y_{nj}. Writing equations like these as matrix multiplications makes them easier to implement in the computer, and often your code will run faster if you do so.

2.2

Negative log-likelihood:

$$-\ln p(\mathbf{T} \mid \mathbf{\Phi}, \mathbf{w}_1, \dots, \mathbf{w}_K) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln y_k(\phi_n) = E(\mathbf{w}_1, \dots, \mathbf{w}_K), \tag{37}$$

where $E(\mathbf{w}_1, \dots, \mathbf{w}_K)$ is the cross-entropy error function for the multi-class classification problem, described in Bishop p.209 Eq. (4.108). The relationship between the cross-entropy error function and the log-likelihood is that they are each other's negative function, i.e. minimizing one is equivalent to maximizing the other. So we could maximize the log-likelihood to find the maximum likelihood, but you would get the same result if you try to find it by minimizing the cross-entropy error function. In the former method we speak of gradient ascent, i.e. weights are updated by adding a certain fraction of the gradient of the log-likelihood. In the latter method, we speak of gradient descent, i.e. weights are updated by subtracting a fraction of the gradient of the error-function.

2.3

• Initialize $\mathbf{w}_1, \dots \mathbf{w}_K$.
• Choose a learning rate $\eta > 0$.
• For $\tau = 1$ to n_B:
  1. Randomly sample B points $\mathbf{x}_1, \dots, \mathbf{x}_B$.
  2. Using Eqs. (36) and (37), update the weights:

$$\mathbf{w}_j^{(\tau+1)} = \mathbf{w}_j^{(\tau)} - \eta \frac{1}{B} \sum_{b=1}^{B} \Big(\nabla_{\mathbf{w}_j} E(\mathbf{x}_b, \mathbf{w}_j) \Big)^\top \quad \text{for each } j \in \{1, \dots, K\}. \tag{38}$$

• Return $\mathbf{w}_1, \dots, \mathbf{w}_K$.

Stochastic gradient descent with single data points is relatively sensitive to variance in the data, while mini-batch gradient descent averages over a larger number of data points, resulting in a “smoother” updating process towards the approximate local minimum of the error function. So for mini-batch gradient descent the steps are more often taken in the right direction than for the single point case, making it more efficient. Full batch gradient descent has even less variance in the gradient, however we can’t always perform full batch gradient descent as it requires an unfeasible amount of memory in some applications.

2.4

$$W_1\mathbf{x} = \begin{bmatrix} 0.4 & 0.87 \\ 0.58 & 0.34 \end{bmatrix} \begin{bmatrix} 0.3 \\ 0.7 \end{bmatrix} = \begin{bmatrix} 0.729 \\ 0.412 \end{bmatrix} := \mathbf{a}_1 = \begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix} \tag{39}$$

$$\tanh(\mathbf{a}_1) = \begin{bmatrix} 0.622 \\ 0.39 \end{bmatrix} := \mathbf{z} \tag{40}$$

$$W_2\mathbf{z} = \begin{bmatrix} 0.12 & 0.87 \\ 0.82 & 0.31 \\ 0.77 & 0.9 \end{bmatrix} \begin{bmatrix} 0.622 \\ 0.39 \end{bmatrix} = \begin{bmatrix} 0.414 \\ 0.631 \\ 0.83 \end{bmatrix} := \mathbf{a}_2 = \begin{bmatrix} a_{12} \\ a_{22} \\ a_{32} \end{bmatrix} \tag{41}$$

$$\text{softmax}(\mathbf{a}_2) = \begin{bmatrix} 0.266 \\ 0.331 \\ 0.403 \end{bmatrix} := \mathbf{y} \tag{42}$$

$$\text{loss} = -\ln(y_3) = 0.909 \tag{43}$$

Derivative of E w.r.t. w_5:

$$\frac{\partial E}{\partial w_5} = \sum_{k=1}^{3} \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial w_5} \tag{44}$$

$$\frac{\partial E}{\partial y_k} = \frac{t_{nk}}{y_k} \tag{45}$$

$t_{nk} = 1$ only for $k = 3$, zero otherwise, so:

$$\frac{\partial E}{\partial w_5} = -\frac{1}{y_3} \frac{\partial y_3}{\partial w_5} \tag{46}$$

$$= -\frac{1}{y_3} \frac{\partial y_3}{\partial a_{12}} \frac{\partial a_{12}}{\partial w_5} \tag{47}$$

$$= -\frac{1}{y_3} \frac{\partial}{\partial a_{12}} \text{softmax}(a_{32}) \frac{\partial}{\partial w_5} (w_5 z_1 + w_6 z_2) \tag{48}$$

$$= -\frac{1}{y_3} \big(-\text{softmax}(a_{12})\text{softmax}(a_{32}) z_1 \big) \tag{49}$$

$$= -\frac{1}{y_3} \big(-y_1 y_3 z_1 \big) \tag{50}$$

$$= y_1 z_1 \tag{51}$$

$$= 0.266 \cdot 0.622 = 0.165 \tag{52}$$

Update rule:

$$\mathbf{w}_j^{(\tau+1)} = \mathbf{w}_j^{(\tau)} - \eta \frac{\partial E}{\partial \mathbf{w}_j}, \tag{53}$$

with $\eta = 0.05$. After updating we get:

$$W_1 = \begin{bmatrix} 0.401 & 0.873 \\ 0.583 & 0.346 \end{bmatrix} \tag{54}$$

$$W_2 = \begin{bmatrix} 0.112 & 0.865 \\ 0.81 & 0.304 \\ 0.789 & 0.912 \end{bmatrix} \tag{55}$$

$$W_1\mathbf{x} = \begin{bmatrix} 0.731 \\ 0.417 \end{bmatrix} \tag{56}$$

$$\tanh(W_1\mathbf{x}) = \begin{bmatrix} 0.624 \\ 0.394 \end{bmatrix} = \mathbf{z} \tag{57}$$

$$W_2\mathbf{z} = \begin{bmatrix} 0.411 \\ 0.625 \\ 0.852 \end{bmatrix} \tag{58}$$

$$\mathbf{y} = \text{softmax}(W_2\mathbf{z}) = \begin{bmatrix} 0.264 \\ 0.327 \\ 0.41 \end{bmatrix} \tag{59}$$

$$\text{loss} = -\ln y_3 = 0.892 \tag{60}$$

The error has decreased after the update.