SHRUTI RAO
2636454
s2.rao@student.vu.nl

Homework Assignment 3

Machine Learning 1, 19/20

2019-10-04

Naive Bayes Classification

1

The data likelihood without any independence assumptions are given by:

$$p(\boldsymbol{T}, \boldsymbol{X}|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(t_n, \boldsymbol{x}_n|\boldsymbol{\theta})$$

Further, using the probability rule for Bayes, the data likelihood for the general k classes naive Bayes classifier is:

$$p(\boldsymbol{T}, \boldsymbol{X}|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(t_n, \boldsymbol{x}_n|\boldsymbol{\theta})$$

Splitting the expression using the product rule:

$$= \prod_{n=1}^{N} p(t_n|\boldsymbol{\theta})p(\boldsymbol{x}_n|t_n, \boldsymbol{\theta})$$

$$= \prod_{n=1}^{N} \prod_{k=1}^{K} p(t_n = k|\boldsymbol{\theta})p(\boldsymbol{x}_n|t_n = k, \boldsymbol{\theta})^{\mathcal{I}(t_n=k)}$$

Assuming the Naive Bayes assumption of product over D:

$$= \prod_{n=1}^{N} \prod_{k=1}^{K} \left(\pi_k \prod_{d=1}^{D} p(\boldsymbol{x}_{nd}|C_k, \theta_{dk})\right)^{\mathcal{I}(t_n=k)}$$

2

The Naive Bayes assumption of conditional independence reduces number of parameters changes to $2n$. The conditional independence assumption is naive because it can rarely be applicable in real life scenarios except for the few cases wherein Naive Bayes performs extremely well. The Naive Bayes assumption will fail when the predictors are dependent on one another, but get ignored by Naive Bayes. For example, when predicting probability of a person being hired, the predictor variables include age, experience, education level. Here experience and age and education level all depend on each other and influence each other. This influence would be overlooked by Naive Bayes. `https://nlp.stanford.edu/IR-book/html/htmledition/properties-of-naive-bayes-1.html`

3

The Bernoulli Distribution is given by:

$$p(\boldsymbol{x}|C_k, \theta_{1k}, ..., \boldsymbol{theta}_{dk}) = \prod_{d=1}^{D} \theta_{dk}^{x_d}(1 - \theta_{dk})^{1-x_{nd}}$$

Rewriting the data likelihood in terms of this distribution, we get:

$$p(\boldsymbol{T}, \boldsymbol{X}|\boldsymbol{\theta}) = \prod_{n=1}^{N} \prod_{k=1}^{K} \left(\pi_k \prod_{d=1}^{D} \theta_{dk}^{x_{dk}}(1 - \theta_{dk})^{1-\boldsymbol{x}_{nd}}\right)$$

The log likelihood is therefore:

$$\begin{aligned}
\ln p(\boldsymbol{T}, \boldsymbol{X}|\boldsymbol{\theta}) &= \ln \prod_{n=1}^{N} \prod_{k=1}^{K} \left(\pi_k \prod_{d=1}^{D} \theta_{dk}^{x_{nd}} (1-\theta_{dk})^{1-\boldsymbol{x}_{nd}} \right) \\
&= \sum_{n=1}^{N} \ln \prod_{k=1}^{K} \left(\pi_k \prod_{d=1}^{D} \theta_{dk}^{x_{nd}} (1-\theta_{dk})^{1-\boldsymbol{x}_{nd}} \right) \text{ [as } ln \prod = \sum] \\
&= \sum_{k=1}^{K} \sum_{n \in C_k}^{N} \ln \left(\pi_k \prod_{d=1}^{D} \theta_{dk}^{x_{nd}} (1-\theta_{dk})^{1-\boldsymbol{x}_{nd}} \right) \\
&= \sum_{k=1}^{K} \sum_{n \in C_k}^{N} \left(\ln \pi_k + \sum_{d=1}^{D} \boldsymbol{x}_{nd} \ln \theta_{dk} + (1-\boldsymbol{x}_{nd}) \ln (1-\theta_{dk}) \right)
\end{aligned}$$

4

$$\ln p(\boldsymbol{T}, \boldsymbol{X}|\boldsymbol{\theta}) = \sum_{k=1}^{K} \sum_{n \in C_k}^{N} \left(\ln \pi_k + \sum_{d=1}^{D} \boldsymbol{x}_{nd} \ln \theta_{dk} + (1-\boldsymbol{x}_{nd}) \ln (1-\theta_{dk}) \right)$$

Taking the derivative:

$$\begin{aligned}
\frac{\partial \ln p(\boldsymbol{T}, \boldsymbol{X}|\boldsymbol{\theta})}{\partial \theta_{dk}} &= \frac{\partial}{\partial \theta_{dk}} \sum_{k=1}^{K} \sum_{n \in C_k}^{N} \left(\ln \pi_k + \sum_{d=1}^{D} \boldsymbol{x}_{nd} \ln \theta_{dk} + (1-\boldsymbol{x}_{nd}) \ln (1-\theta_{dk}) \right) \\
0 &= 0 + \sum_{n \in C_k}^{N} \left[\frac{\boldsymbol{x}_{nd}}{\theta_{dk}} - \frac{(1-\boldsymbol{x}_{nd})}{(1-\theta_{dk})} \right] \\
0 &= \sum_{n \in C_k}^{N} \left[\frac{\boldsymbol{x}_{nd}}{\theta_{dk}} \right] - \sum_{n \in C_k}^{N} \left[\frac{(1-\boldsymbol{x}_{nd})}{(1-\theta_{dk})} \right] \\
\sum_{n \in C_k}^{N} \left[\frac{\boldsymbol{x}_{nd}}{\theta_{dk}} \right] &= \sum_{n \in C_k}^{N} \left[\frac{(1-\boldsymbol{x}_{nd})}{(1-\theta_{dk})} \right] \\
\sum_{n \in C_k}^{N} \left[\frac{(1-\theta_{dk})}{\theta_{dk}} \right] &= \sum_{n \in C_k}^{N} \left[\frac{(1-\boldsymbol{x}_{nd})}{\boldsymbol{x}_{nd}} \right] \\
\sum_{n \in C_k}^{N} \left[\frac{1}{\theta_{dk}} - \frac{\theta_{dk}}{\theta_{dk}} \right] &= \sum_{n \in C_k}^{N} \left[\frac{1}{\boldsymbol{x}_{nd}} - \frac{\boldsymbol{x}_{nd}}{\boldsymbol{x}_{nd}} \right] \\
\sum_{n \in C_k}^{N} \left[\frac{1}{\theta_{dk}} \right] - \sum_{n \in C_k}^{N} \left[\frac{\theta_{dk}}{\theta_{dk}} \right] &= \sum_{n \in C_k}^{N} \left[\frac{1}{\boldsymbol{x}_{nd}} \right] - \sum_{n \in C_k}^{N} \left[\frac{\boldsymbol{x}_{nd}}{\boldsymbol{x}_{nd}} \right] \\
\sum_{n \in C_k}^{N} \left[\frac{1}{\theta_{dk}} \right] - \sum_{n \in C_k}^{N} [1] &= \sum_{n \in C_k}^{N} \left[\frac{1}{\boldsymbol{x}_{nd}} \right] - \sum_{n \in C_k}^{N} [1] \\
\sum_{n \in C_k}^{N} \left[\frac{1}{\theta_{dk}} \right] &= \sum_{n \in C_k}^{N} \left[\frac{1}{\boldsymbol{x}_{nd}} \right] \\
\sum_{n \in C_k}^{N} \theta_{dk} &= \sum_{n \in C_k}^{N} \boldsymbol{x}_{nd}
\end{aligned}$$

solving for θ_{dk}:

$$\theta_{dk} = \frac{1}{N_k} \sum_{n \in C_k}^{N} \boldsymbol{x}_{nd}$$

Here, θ_{dk} is represents the average number of words *d* for each document, for class *k*.

5

$$
\begin{aligned}
p(\boldsymbol{\mathcal{C}_1}|\boldsymbol{x}) &= \frac{p(\boldsymbol{x}|\boldsymbol{\mathcal{C}_1})p(\boldsymbol{\mathcal{C}_1})}{p(\boldsymbol{x})} \\
&= \frac{p(\boldsymbol{x}|\boldsymbol{\mathcal{C}_1})p(\boldsymbol{\mathcal{C}_1})}{\sum_{k=1}^{K} p(\boldsymbol{x_k}|\boldsymbol{\mathcal{C}_k})p(\boldsymbol{\mathcal{C}_k})}
\end{aligned}
$$

$$\text{Using } \; p(x|\mathcal{C}_k) = \prod_{d=1}^{D} p(x_d|\mathcal{C}_k) \; :$$

$$= \frac{\pi_1 \prod_{d=1}^{D} p(\boldsymbol{x_{nd}}|\boldsymbol{\theta_{d1}})}{\sum_{k=1}^{K} \pi_k \prod_{d=1}^{D} p(\boldsymbol{x_{nd}}|\theta_{dk})}$$

6

The Bernoulli Distribution is given by:

$$p(\boldsymbol{x}) = \pi\theta_{dk}^{x_d}(1-\theta_{dk})^{1-x_{nd}}$$

Thus, rewriting it, we get:

$$
\begin{aligned}
p(\boldsymbol{\mathcal{C}_1}|\boldsymbol{x}) &= \frac{p(\boldsymbol{x}|\boldsymbol{\mathcal{C}_1})p(\boldsymbol{\mathcal{C}_1})}{p(\boldsymbol{x})} \\
&= \frac{\pi_1 \prod_{d=1}^{D} \left[\theta_{d1}^{x_d}(1-\theta_{d1})^{1-x_d}\right]}{\sum_{k=1}^{K} \pi_k \prod_{d=1}^{D} \left[\theta_{dk}^{x_d}(1-\theta_{dk})^{1-x_d}\right]}
\end{aligned}
$$

Multi-class Logistic Regression and Multilayer Perceptrons

1.

Likelihood:

$$
\begin{aligned}
p(\boldsymbol{T}|\boldsymbol{\Phi}, \boldsymbol{w_1}, ..., \boldsymbol{w_k}) &= \prod_{n=1}^{N}\prod_{k=1}^{K} p(\boldsymbol{\mathcal{C}}_k|\Phi_n)^{t_{nk}} \\
&= \prod_{n=1}^{N}\prod_{k=1}^{K} y_{nk}^{t_{nk}} \\
&= \prod_{n=1}^{N}\prod_{k=1}^{K} y_k(\Phi_n)^{t_{nk}}
\end{aligned}
$$

Here **T** is a target matrix (N x K) with elements t_{nk}

Log-Likelihood:

$$
\begin{aligned}
\ln p(\boldsymbol{T}|\boldsymbol{\Phi}, \boldsymbol{w_1}, ..., \boldsymbol{w_k}) &= \ln \prod_{n=1}^{N}\prod_{k=1}^{K} y_k(\Phi_n)^{t_{nk}} \\
&= \sum_{n=1}^{N} \ln \prod_{k=1}^{K} y_k(\Phi_n)^{t_{nk}} \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_k(\Phi_n)
\end{aligned}
$$

Derivative $\frac{\partial y_k}{\partial w_j}$:

Assuming the convention that $\frac{\partial f(x)}{\partial x} = (\frac{\partial f(x)}{\partial x_1}, ..., \frac{\partial f(x)}{\partial x_D})$ is a row vector that will be used against the column vector x, thus the corresponding gradient vector is again a row vector as well.
Also noting that: $\frac{\partial y_k}{\partial w_j} = \frac{\partial exp(a_k)}{\sum_i exp(a_i)}$

$$= \frac{\sum_i exp(a_i).exp(a_k)\frac{\partial a_k}{\partial w_j} - exp(a_k)\sum_i exp(a_i)\frac{\partial a_i}{\partial w_j}}{[\sum_i exp(a_i)]^2}$$
$$= \frac{\sum_i exp(a_i).exp(a_k)\frac{\partial a_k}{\partial w_j}}{[\sum_i exp(a_i)]^2} - \frac{exp(a_k)\sum_i exp(a_i)\frac{\partial a_i}{\partial w_j}}{[\sum_i exp(a_i)]^2}$$
$$= \frac{exp(a_k)\frac{\partial a_k}{\partial w_j}}{[\sum_i exp(a_i)]} - \frac{exp(a_k)\sum_i exp(a_i)\frac{\partial a_i}{\partial w_j}}{[\sum_i exp(a_i)]^2}$$

As $a_k = w_k^T \Phi$ and $\dfrac{\partial y_k}{\partial a_j} = y_k[I_{kj} - y_j]$

$$= y_k(\Phi)\Phi^T I_{kj} - y_j y_k(\Phi)\Phi^T$$
$$= y_k(\Phi)\Phi^T[I_{kj} - y_j(\Phi)]$$

Compute $\nabla_{w_j} \ln p(\boldsymbol{T}|\boldsymbol{\Phi}, \boldsymbol{w_1}, ..., \boldsymbol{w_K})$:

From above: $\frac{\partial y_k}{\partial w_j} = y_k(\Phi)\Phi^T[I_{kj} - y_j(\Phi)]$

$$\nabla_{w_j} \ln p(\boldsymbol{T}|\boldsymbol{\Phi}, \boldsymbol{w_1}, ..., \boldsymbol{w_K}) = \sum_{n=1}^{N}\sum_{k=1}^{K}[t_{nk}\ln y_k(\Phi_n)]\frac{\partial y_k}{\partial w_j}$$
$$= \sum_{n=1}^{N}\sum_{k=1}^{K}\left[\frac{t_{nk}}{y_k(\Phi_n)}\right][y_k(\Phi_n)\Phi_n^T[I_{kj} - y_j(\Phi_n)])]$$
$$= \sum_{n=1}^{N}\sum_{k=1}^{K}\left[\frac{t_{nk}}{y_k(\Phi_n)}y_k(\Phi_n)\Phi_n^T I_{kj} - \frac{t_{nk}}{y_k(\Phi_n)}y_k(\Phi_n)\Phi_n^T y_j(\Phi_n)\right]$$
$$= \sum_{n=1}^{N}\sum_{k=1}^{K}\left[t_{nk}\Phi_n^T I_{kj} - t_{nk}\Phi_n^T y_j(\Phi_n)\right]$$
$$= \sum_{n=1}^{N}\sum_{k=1}^{K}\left[t_{nk}\Phi_n^T I_{kj}\right] - \sum_{n=1}^{N}\sum_{k=1}^{K}\left[t_{nk}\Phi_n^T y_j(\Phi_n)\right]$$
$$= \sum_{n=1}^{N}\Phi_n^T\sum_{k=1}^{K}t_{nk}I_{kj} - \sum_{n=1}^{N}\Phi_n^T y_j(\Phi_n)\sum_{k=1}^{K}t_{nk}$$

From Bishop $\displaystyle\sum_{k=1}^{K} t_{nk} = 1$

Summation of identity matrix times t is another t_{nj}matrix

$$= \sum_{n=1}^{N}\Phi_n^T t_{nj} - \sum_{n=1}^{N}\Phi_n^T y_j(\Phi_n)$$
$$= \sum_{n=1}^{N}\Phi_n^T\left[t_{nj} - y_j(\Phi_n)\right]$$

Your gradient is now a sum, can you rewrite as a product? and Why is it useful?

Yes it can be written as a matrix multiplication. Why is it useful?

2

The negative log likelihood will be:

$$\begin{aligned}-\ln p(\boldsymbol{T}|\boldsymbol{\Phi}, \boldsymbol{w_1}, ..., \boldsymbol{w_k}) &= -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\ln y_k(\Phi_n) \\ &= E(w_1, ..., w_k)\end{aligned}$$

Thus, taking the negative log gives the cross entropy error function. The relationship between the cross entropy error function and the log likelihood function would be inverse due to the presence of the negative sign. So, maximizing the log likelihood would minimize the cross entropy error. The weights get updated based on the equation: $w^{(\tau+1)} = w^{\tau} - \eta\nabla E_n$ and with Cross Entropy, the E_n has a negative sign that gets cancelled out hence there is no effect on the weight updates during optimization (http://www.awebb.info/blog/cross_entropy.

3

Initialize batch size to be B
Initialize the number of batches $= n_B$
$E_n = -\sum_{k=1}^{K} t_{nk}\ln y_k(\Phi_n)$ Initialize $t = 0$ where $t \in 0, .., T$
while t is less than or equal to T for mini batch in list of mini-batch $[n_1, ..., n_B]$

1. Compute forward pass
   - make prediction
   - compute error
2. Compute backward pass
   - Compute gradient $\frac{\partial E}{\partial W_{ij}}$
   - Update weights using $w^{(\tau+1)} = w^{\tau} - \eta\nabla E_n$

Mini batch gradient can be more computationally effective when using extremely large datasets. As the model gets updated at the end of one training epoch. The work can also be distributed across machines and performed in parallel allowing for much faster speeds in computation.(https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/)

4

$x = (0.3, 0.7)^T$
$y = (0, 0, 1)^T$
To compute the forward pass:

$$
\begin{aligned}
a_0 &= W_1 x \\
&= \begin{bmatrix} 0.4 & 0.87 \\ 0.58 & 0.34 \end{bmatrix} \begin{bmatrix} 0.3 \\ 0.7 \end{bmatrix} \\
&= \begin{bmatrix} 0.729 \\ 0.412 \end{bmatrix} \\
z_0 &= tanh(a_0) \\
&= \frac{e^z - e^{-z}}{e^z + e^{-z}} \\
&= \begin{bmatrix} \frac{e^{0.729} - e^{-0.729}}{e^{0.729} + e^{-0.729}} \\ \frac{e^{0.412} - e^{-0.412}}{e^{0.412} + e^{-0.412}} \end{bmatrix} \\
&= \begin{bmatrix} 0.622 \\ 0.390 \end{bmatrix} \\
a_1 &= W_2 z_0 \\
&= \begin{bmatrix} 0.12 & 0.87 \\ 0.82 & 0.31 \\ 0.77 & 0.9 \end{bmatrix} \begin{bmatrix} 0.622 \\ 0.390 \end{bmatrix} \\
&= \begin{bmatrix} 0.414 \\ 0.630 \\ 0.830 \end{bmatrix} \\
y_{out} &= softmax(a_1) \\
&= \frac{exp(a_1)}{\sum_{j=1}^{3} exp(a_j)} \\
&= \begin{bmatrix} \frac{exp(0.414)}{exp(0.414)+exp(0.630)+exp(0.830)} \\ \frac{exp(0.630)}{exp(0.414)+exp(0.630)+exp(0.830)} \\ \frac{exp(0.830)}{exp(0.414)+exp(0.630)+exp(0.830)} \end{bmatrix} \\
&= \begin{bmatrix} 0.266 \\ 0.330 \\ 0.403 \end{bmatrix}
\end{aligned}
$$

The cross entropy loss error E is then given by:

$$
\begin{aligned}
E(w_1, ...w_K) &= -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln y_k \\
&= -(0. \ln 0.266 + 0. \ln 0.330 + 1. \ln 0.403) \\
&= 0.908
\end{aligned}
$$

Computing the derivative $\frac{\partial E}{\partial w_5}$:

From Bishop 5.3.2, The derivative for the first layer weight $\frac{\partial E}{\partial w_5}$ is given by:

$$
\frac{\partial E}{\partial w_5} = \sum_{i=1} \frac{\partial E}{\partial y_{out}} * \frac{\partial y_{out}}{\partial a_1} * \frac{\partial a_1}{\partial w_5}
$$

(From Bishop 5.3.2) This can be reduced to :

$$\begin{aligned}
\frac{\partial E}{\partial w_5} &= \delta_{01} * z_0^1 \\
\delta_{01} &= -(y_{01} - y_{out})(y_{out})(1 - y_{out}) \\
&= -(0 - 0.266)(0.266)(1 - 0.266) \\
&= 0.052 \\
\frac{\partial E}{\partial w_5} &= 0.052 * 0.622 \\
&= 0.032
\end{aligned}$$

Weight Update with $\eta = 0.05$:

$\frac{\partial E}{\partial w_5} = 0.032$ Weight Update is given by:

$$\begin{aligned}
w^{(\tau+1)} &= w^{\tau} - \eta \nabla E_n \\
w_5^{(\tau+1)} &= w_5^{\tau} - (0.05)\nabla E_5 \\
&= 0.12 - (0.05)(0.032) \\
&= 0.118
\end{aligned}$$

Similarly, the weights for $w_1, w_2, w_3, w_4, w_6, w_7, w_8, w_9, w_{10}$ can be calculated:

$$\begin{aligned}
w_1^{(\tau+1)} &= 0.4 - (0.05 * -0.029) = 0.401 \\
w_2^{(\tau+1)} &= 0.87 - (0.05 * -0.067) = 0.873 \\
w_3^{(\tau+1)} &= 0.58 - (0.05 * -0.052) = 0.583 \\
w_4^{(\tau+1)} &= 0.34 - (0.05 * -0.120) = 0.346 \\
w_6^{(\tau+1)} &= 0.87 - (0.05 * 0.104) = 0.865 \\
w_7^{(\tau+1)} &= 0.82 - (0.05 * 0.206) = 0.810 \\
w_8^{(\tau+1)} &= 0.31 - (0.05 * 0.129) = 0.304 \\
w_9^{(\tau+1)} &= 0.77 - (0.05 * -0.371) = 0.789 \\
w_{10}^{(\tau+1)} &= 0.9 - (0.05 * -0.233) = 0.912
\end{aligned}$$

Therefore the updated weights for W_1 and W_2 are:

$$\begin{aligned}
W_1 &= \begin{bmatrix} 0.401 & 0.873 \\ 0.583 & 0.346 \end{bmatrix} \\
W_2 &= \begin{bmatrix} 0.118 & 0.865 \\ 0.810 & 0.304 \\ 0.789 & 0.912 \end{bmatrix}
\end{aligned}$$

Compute Loss

$$
\begin{aligned}
a_0 &= W_1 x \\
&= \begin{bmatrix} 0.401 & 0.873 \\ 0.583 & 0.346 \end{bmatrix} \begin{bmatrix} 0.3 \\ 0.7 \end{bmatrix} \\
&= \begin{bmatrix} 0.731 \\ 0.417 \end{bmatrix} \\
z_0 &= tanh(a_0) \\
&= \frac{e^z - e^{-z}}{e^z + e^{-z}} \\
&= \begin{bmatrix} \frac{e^{0.731} - e^{-0.731}}{e^{0.731} + e^{-0.731}} \\ \frac{e^{0.417} - e^{-0.417}}{e^{0.417} + e^{0.417}} \end{bmatrix} \\
&= \begin{bmatrix} 0.624 \\ 0.394 \end{bmatrix} \\
a_1 &= W_2 z_0 \\
&= \begin{bmatrix} 0.118 & 0.865 \\ 0.810 & 0.304 \\ 0.789 & 0.912 \end{bmatrix} \begin{bmatrix} 0.624 \\ 0.394 \end{bmatrix} \\
&= \begin{bmatrix} 0.414 \\ 0.625 \\ 0.852 \end{bmatrix} \\
y_{out} &= softmax(a_1) \\
&= \begin{bmatrix} \frac{exp(0.414)}{exp(0.414)+exp(0.625)+exp(0.852)} \\ \frac{exp(0.625)}{exp(0.414)+exp(0.625)+exp(0.852)} \\ \frac{exp(0.852)}{exp(0.414)+exp(0.625)+exp(0.852)} \end{bmatrix} \\
&= \begin{bmatrix} 0.264 \\ 0.326 \\ 0.409 \end{bmatrix}
\end{aligned}
$$

The cross entropy loss error E is now given by:

$$
\begin{aligned}
E(w_1, ...w_K) &= -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln y_k \\
&= -(0. \ln 0.264 + 0. \ln 0.326 + 1. \ln 0.409) \\
&= 0.894
\end{aligned}
$$

The error now decreased to 0.894 from 0.909, indicating that the backpropagation helped in correcting the error by changing the weights in the desired manner.