Machine Learning 1 - Practice exercise 3

1 Naive Bayes Spam Classification

Naive Bayes (NB) is a particular form of classification that makes strong independence assumptions regarding the features of the data, conditional on the classes (see Bishop section 4.2.3). Specifically, NB assumes each feature is independent given the class label. In contrast, when we looked at probabilistic generative models for classification in the lecture, we used a full-covariance Gaussian to model data from each class, which incorporates correlation between all the input features (i.e. they are not conditionally independent).

If correlated features are treated independently, the evidence for a class will be overcounted. However, Naive Bayes is very simple to construct, because by ignoring correlations the *class-conditional likelihood*, $p(\mathbf{x}|\mathcal{C}_k)$, is a product of D univariate distributions, each of which is simple to learn:

$$p(\mathbf{x}|\mathcal{C}_k) = \prod_{d=1}^{D} p(x_d|\mathcal{C}_k) \tag{1}$$

Consider a spam filter that classifies your emails into three classes $\mathcal{C}_1$ (spam), $\mathcal{C}_2$ (non-spam) and $\mathcal{C}_3$ (ambiguous). If the classifier is not certain whether the email is spam or non-spam, it will assign it to the $\mathcal{C}_3$ (ambiguous) case, indicating it needs help from humans. To do this you first make a *bag-of-words* (BoW) representation of your entire training set (a bunch of spam emails and non-spam emails). A BoW is a vector of dimension D of word counts, one for each document (i.e. the words go into a bag and are shaken, losing their order so only their count matters). You can think of D as the vocabulary size of the training set, but it may also contain tokens or special features you think are important for spam detection. Your training set is therefore consists of an N by D matrix of word counts $\mathbf{X}$, and the target vector $\mathbf{t} = (t_1, ..., t_N)^T$, such that $t_n = 1$, if $n \in \mathcal{C}_1$, and $t_n = 2$, if $n \in \mathcal{C}_2$ and $t_n = 3$ if $n \in \mathcal{C}_3$. Assume we know $p(\mathcal{C}_1) = \pi_1$, $p(\mathcal{C}_2) = \pi_2$ and $p(\mathcal{C}_3) = \pi_3$ (with the constraint $\pi_1 + \pi_2 + \pi_3 = 1$). We can model the word counts using

different distributions, but for this question we will use a Poisson distribution model:

$$\begin{aligned} p(x_d|\mathcal{C}_k, \theta_{dk}) &= \mathcal{P}\left(x_d|\lambda_{dk}\right) \\ &= \frac{\lambda_{dk}^{x_d}}{x_d!}\exp(-\lambda_{dk}) \end{aligned}$$

with distribution parameters $\theta_{dk} = \lambda_{dk}$.

With this information answer the following questions:

1. Write down the data likelihood, $p(\mathbf{T}, \mathbf{X}|\boldsymbol{\theta})$, for the *general* three class naive Bayes classifier. Use the indicator function $\mathbb{I}(x = a) = \begin{cases} 1 & \text{if} \quad x = a \\ 0 & \text{otherwise} \end{cases}$. You should write the likelihood in terms of $p(x_d|\mathcal{C}_k, \theta_{dk})$, meaning you should not assume the explicit Poisson distribution.

Solutions

4 points: 1 for product over N datapoints, 1 for correct use of indicator function, 1 for correct use of product rule with prior and class conditional densities, 1 for correct use of Naive Bayes assumption (product over d).

$$
\begin{aligned}
p(\mathbf{T}, \mathbf{X}|\boldsymbol{\theta}) &= \prod_{n=1}^{N} p(t_n, \mathbf{x}_n|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(t_n|\boldsymbol{\theta}) p(\mathbf{x}_n|t_n, \boldsymbol{\theta}) \\
&= \prod_{n=1}^{N} \Big(p(t_n = 1|\boldsymbol{\theta}) p(\mathbf{x}_n|t_n = 1, \boldsymbol{\theta}) \Big)^{\mathbb{I}(t_n=1)} \\
&\qquad \cdot \Big(p(t_n = 2|\boldsymbol{\theta}) p(\mathbf{x}_n|t_n = 2, \boldsymbol{\theta}) \Big)^{\mathbb{I}(t_n=2)} \\
&\qquad \cdot \Big(p(t_n = 3|\boldsymbol{\theta}) p(\mathbf{x}_n|t_n = 3, \boldsymbol{\theta}) \Big)^{\mathbb{I}(t_n=3)} \\
&= \prod_{n=1}^{N} \left(\pi_1 \prod_{d=1}^{D} p(x_{nd}|\mathcal{C}_1, \theta_{d1}) \right)^{\mathbb{I}(t_n=1)} \\
&\qquad \cdot \left(\pi_2 \prod_{d=1}^{D} p(x_{nd}|\mathcal{C}_2, \theta_{d2}) \right)^{\mathbb{I}(t_n=2)} \\
&\qquad \cdot \left(\pi_3 \prod_{d=1}^{D} p(x_{nd}|\mathcal{C}_3, \theta_{d3}) \right)^{\mathbb{I}(t_n=3)}
\end{aligned}
$$

2. Write down the data likelihood $p(\mathbf{T}, \mathbf{X}|\boldsymbol{\theta})$ for the Poisson model.

Solutions

1 point: correct insertion of Poisson distribution.

$$p(\mathbf{T}, \mathbf{X}|\boldsymbol{\theta}) = \prod_{n=1}^{N} \left(\pi_1 \prod_{d=1}^{D} \frac{\lambda_{d1}^{x_{nd}}}{x_{nd}!} \exp(-\lambda_{d1}) \right)^{\mathbb{I}(t_n=1)} \cdot \left(\pi_2 \prod_{d=1}^{D} \frac{\lambda_{d2}^{x_{nd}}}{x_{nd}!} \exp(-\lambda_{d2}) \right)^{\mathbb{I}(t_n=2)} \cdot \left(\pi_3 \prod_{d=1}^{D} \frac{\lambda_{d3}^{x_{nd}}}{x_{nd}!} \exp(-\lambda_{d3}) \right)^{\mathbb{I}(t_n=3)}$$

3. Write down the log-likelihood for the Poisson model.

Solutions

2 points: correct use of products and sums for logarithms.

$$\begin{aligned} \ln p(\mathbf{T}, \mathbf{X}|\boldsymbol{\theta}) = & \sum_{n \in \mathcal{C}_1}^{N} \left(\ln \pi_1 + \sum_{d=1}^{D} x_{nd} \ln \lambda_{d1} - \ln(x_{nd}!) - \lambda_{d1} \right) \\ & + \sum_{n \in \mathcal{C}_2}^{N} \left(\ln \pi_2 + \sum_{d=1}^{D} x_{nd} \ln \lambda_{d2} - \ln(x_{nd}!) - \lambda_{d2} \right) \\ & + \sum_{n \in \mathcal{C}_3}^{N} \left(\ln \pi_3 + \sum_{d=1}^{D} x_{nd} \ln \lambda_{d3} - \ln(x_{nd}!) - \lambda_{d3} \right) \end{aligned} \tag{2}$$

Note. Here you need to use $\ln \prod = \sum \ln$, however, keep in mind that $\ln \sum \prod \neq \sum \ln \prod$

4. Solve for the MLE estimators for λ_{dk}.

Solutions

5 points: 3 for correct derivative, 1 for setting equal to 0 for ML estimate, and 1 for correct solving for ML estimate.

$$\frac{\partial \ln p(\mathbf{T}, \mathbf{X}|\boldsymbol{\theta})}{\partial \lambda_{dk}} = \sum_{n \in \mathcal{C}_k}^{N} \left(\frac{x_{nd}}{\lambda_{dk}} - 1 \right) = 0$$

$$\sum_{n \in \mathcal{C}_k}^{N} \frac{x_{nd}}{\lambda_{dk}} = \sum_{n \in \mathcal{C}_k}^{N} 1$$

$$\lambda_{dk} = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k}^{N} x_{nd}$$

I.e. λ_{dk} is the average number of words/tokens d per email for class k.

5. Write $p(\mathcal{C}_1|\mathbf{x})$ for the *general* three class naive Bayes classifier.

Solutions

3 points: 1 for Bayes rule, 1 for denominator, 1 for product over d for Naive Bayes assumption.

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2) + p(\mathbf{x}|\mathcal{C}_3)p(\mathcal{C}_3)}$$

$$= \frac{\pi_1 \prod_{d=1}^{D} p(x_{nd}|\theta_{d1})}{\pi_1 \prod_{d=1}^{D} p(x_{nd}|\theta_{d1}) + \pi_2 \prod_{d=1}^{D} p(x_{nd}|\theta_{d2}) + \pi_3 \prod_{d=1}^{D} p(x_{nd}|\theta_{d3})}$$

6. Write $p(\mathcal{C}_1|\mathbf{x})$ for the Poisson model.

Solutions

1 point for correct insertion of Poisson distributions. The canceling of count factorials is not necessary for a correct answer.

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{\pi_1 \prod_{d=1}^{D} \lambda_{d1}^{x_{nd}} \exp(-\lambda_{d1})}{\pi_1 \prod_{d=1}^{D} \lambda_{d1}^{x_{nd}} \exp(-\lambda_{d1}) + \pi_2 \prod_{d=1}^{D} \lambda_{d2}^{x_{nd}} \exp(-\lambda_{d2}) + \pi_3 \prod_{d=1}^{D} \lambda_{d3}^{x_{nd}} \exp(-\lambda_{d3})}$$

Notice the product over count factorials containing $\frac{1}{x_{nd}!}$ cancels because it is a constant for each class.

7. For the Poisson model, express the conditions (inequalities) of the region where $\mathbf{x}$ is predicted to be in $\mathcal{C}_1$. Make each inequalities as linear inequalities such as $\mathbf{x}^T\mathbf{a} > c$

Solutions

5 points: 2 points for the two inequalities of posterior class probabilities, 1 point for inserting poisson distributions, 2 points for rewriting of into linear equations.

$$p(\mathcal{C}_1|\mathbf{x}) > p(\mathcal{C}_2|\mathbf{x}) \quad \text{and} \quad p(\mathcal{C}_1|\mathbf{x}) > p(\mathcal{C}_3|\mathbf{x})$$

$$\Downarrow$$

$$\pi_1 \prod_{d=1}^{D} \lambda_{d1}^{x_d} \exp(-\lambda_{d1}) > \pi_2 \prod_{d=1}^{D} \lambda_{d2}^{x_d} \exp(-\lambda_{d2}) \quad \text{and}$$

$$\pi_1 \prod_{d=1}^{D} \lambda_{d1}^{x_d} \exp(-\lambda_{d1}) > \pi_3 \prod_{d=1}^{D} \lambda_{d3}^{x_d} \exp(-\lambda_{d3})$$

$$\Downarrow$$

$$\prod_{d=1}^{D} \left(\frac{\lambda_{d1}}{\lambda_{d2}}\right)^{x_d} > \frac{\pi_2}{\pi_1} \prod_{d=1}^{D} \exp(\lambda_{d1} - \lambda_{d2}) \quad \text{and} \quad \prod_{d=1}^{D} \left(\frac{\lambda_{d1}}{\lambda_{d3}}\right)^{x_d} > \frac{\pi_3}{\pi_1} \prod_{d=1}^{D} \exp(\lambda_{d1} - \lambda_{d3})$$

$$\Downarrow$$

$$\sum_{d=1}^{D} x_d \ln\left(\frac{\lambda_{d1}}{\lambda_{d2}}\right) > \underbrace{\ln\left(\frac{\pi_2}{\pi_1}\right) + \sum_{d=1}^{D} \lambda_{d1} - \lambda_{d2}}_{c_{12}} \quad \text{and}$$

$$\sum_{d=1}^{D} x_d \ln\left(\frac{\lambda_{d1}}{\lambda_{d3}}\right) > \underbrace{\ln\left(\frac{\pi_3}{\pi_1}\right) + \sum_{d=1}^{D} \lambda_{d1} - \lambda_{d3}}_{c_{13}}$$

$$\Downarrow$$

$$\mathbf{x}^T \mathbf{a}_{12} > c_{12} \quad \text{where} \quad \mathbf{a}_{12} = (\ln\left(\frac{\lambda_{11}}{\lambda_{12}}\right), \ln\left(\frac{\lambda_{21}}{\lambda_{22}}\right), \cdots, \ln\left(\frac{\lambda_{D1}}{\lambda_{D2}}\right))^T$$

$$\mathbf{x}^T \mathbf{a}_{13} > c_{13} \quad \text{where} \quad \mathbf{a}_{13} = (\ln\left(\frac{\lambda_{11}}{\lambda_{13}}\right), \ln\left(\frac{\lambda_{21}}{\lambda_{23}}\right), \cdots, \ln\left(\frac{\lambda_{D1}}{\lambda_{D3}}\right))^T$$

8. Is the region where $\mathbf{x}$ is predicted to be in $\mathcal{C}_1$ convex? Why?

   Solutions

   2 points

   Yes: It is the intersection of two regions bounded by hyperplanes (convex region).

9. Give a concrete example with a specific application where it is helpful to make algorithms ask humans' help for ambiguous predictions.

Solutions

1 point

- Medical Diagnosis : Especially, false negative is dangerous in medical diagnosis. To reduce false negative rate, for uncertain cases, algorithm can ask human experts assist.
- Autonomous Driving : In foggy weather, object (pedestrian) recognition may operate less reliably. In this case, self-driving car entrusts driving to human drivers or asks information about ambiguous objects.
- etc $\cdots$

2 Multi-class Logistic Regression

In class we saw the binary classification version of logistic regression. Here you will derive the gradients for the general case $K > 2$. Much of the preliminaries are in Bishop 4.3.4. This will be useful for Lab 2.

For $K > 2$ the posterior probabilities take a generalized form of the sigmoid called the softmax:

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

where $a_k = \mathbf{w}_k^T \boldsymbol{\phi}$ and $\boldsymbol{\phi}$ is short for $\boldsymbol{\phi}(\mathbf{x}) = (\phi_0(\mathbf{x}), \phi_1(\mathbf{x}), ..., \phi_{M-1}(\mathbf{x}))^T$ 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 labeling 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 datapoint n belongs to class $\mathcal{C}_k$.

Answer the following questions:

1. Derive $\frac{\partial y_k}{\partial \mathbf{w}_j}$. Bishop uses an indicator function I_{kj}, which you can also view as the element at position (k, j) of the identity matrix; previously we used $\mathbb{I}[k = j]$—they are the same thing.

Solutions

3 points
Note that we will use the convention that $\frac{\partial y_k(\boldsymbol{\phi})}{\partial \boldsymbol{\phi}} = \left(\frac{\partial y_k(\boldsymbol{\phi})}{\partial \phi_0}, ..., \frac{\partial y_k(\boldsymbol{\phi})}{\partial \phi_{M-1}}\right)$ and thus a row vector. No points should be deducted for the column vector convention as long as it is used consistently.

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

2. Write down the likelihood as a product over N and K then write down the log-likelihood. Use the entries of $\mathbf{T}$ as selectors of the correct class.

Solutions

3 points: 2 for correct likelihood, 1 for correct log likelihood.

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

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

3. Derive the gradient of the log-likelihood with respect to $\mathbf{w}_j$.

Solutions

4 points: 1 for using chain rule, 2 for the two identities shown with underbraces, 1 for final answer.

Note that we assume the convention that $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \left(\frac{\partial f(\mathbf{x})}{\partial x_1}, ..., \frac{\partial f(\mathbf{x})}{\partial x_D}\right)$ (a row vector) for a column vector $\mathbf{x}$ of size D and a scalar function $f(\mathbf{x})$ that takes $\mathbf{x}$ as an argument. The gradient of a scalar function with respect to a column vector is again a row vector, so in this case $\nabla f(\mathbf{x}) = \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\right)$.

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

The gradient is thus equal to

$$\nabla_{\mathbf{w}_j} \ln p(\mathbf{T}|\mathbf{\Phi}, \mathbf{w}_1, ..., \mathbf{w}_K) = \left(\frac{\partial \ln p(\mathbf{T}|\mathbf{\Phi}, \mathbf{w}_1, ..., \mathbf{w}_K)}{\partial \mathbf{w}_j}\right)$$

4. What is the objective function we minimize that is equivalent to maximizing the log-likelihood?

Solutions

1 point for negative log-likelihood.

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

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

Sometimes we may write y_{nk} for $y_k(\boldsymbol{\phi}_n)$ to give more clutter-free solutions. Minimizing the cross-entropy requires the same gradients as maximizing the log-likelihood, except there is a change in sign:

$$\begin{aligned}\frac{\partial E(\mathbf{w}_1, ..., \mathbf{w}_K)}{\partial \mathbf{w}_j} &= \sum_{n=1}^{N}(y_{nj} - t_{nj})\boldsymbol{\phi}_n \\ &= \sum_{n=1}^{N}\mathbf{e}_n\end{aligned}$$

5. 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.

Solutions

5 points: 1 for each part a, b, c.i, c.ii, d. The step shown in c.iii (decreasing learning rate) is optional, no points should be deducted if absent.

The single step update for SGD is:

$$\mathbf{w}_j^{t+1} = \mathbf{w}_j^t - \eta^t (\nabla \mathbf{e}_n)^T$$

(a) Initialize $\mathbf{w}_1, ..., \mathbf{w}_K$

(b) Initialize η

(c) For $t = 1$ to T do:

i. Randomly choose n from $[1, N]$

ii. $\mathbf{w}_j = \mathbf{w}_j - \eta(\nabla \mathbf{e}_n)^T$ (for all j)

iii. Decrease η

(d) Return $\mathbf{w}_1, ..., \mathbf{w}_K$

6. (**Bonus**) 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.

Solutions

1 point

- 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 ..