Homework 2 - Introduction to Machine Learning for Engineers

kipngeno koech - bkoech

February 12, 2025

1 Naive Bayes Parameters

Problem Statement

The naïve Bayes approach assumes that the feature vectors are independent given the label, that is, for any given data point $\mathbf{x}_i = [x_{i1}, x_{i2}, \cdots, x_{id}]^\top$ and its label y_i, we have

$$P(x_{i1}, x_{i2}, \cdots, x_{id}, y_i) = P(y_i) \prod_{j=1}^{d} P(x_{ij} \mid y_i)$$

Suppose that we are given the data set $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]^\top$, where $\mathbf{X} \in \mathbb{R}^{N\times 4}$, consisting of N data points and 4 features, and its label $\mathbf{y} = [y_1, y_2, \cdots, y_N]^\top$. Assume that the first feature has 2, the second has 3, the third has 4, and the fourth has 5 possible values: Let $X_1 \in \{1, 2\}$, $X_2 \in \{1, 2, 3\}$, $X_3 \in \{1, 2, 3, 4\}$, and $X_4 \in \{1, 2, 3, 4, 5\}$. Also, there are 4 possible labels, i.e., $Y \in \{1, 2, 3, 4\} \forall i$.

a. [5 points] Determine the number of free parameters θ and π that one has to estimate using the naïve Bayes framework. (We use θ for the probabilities of a feature given a label (e.g., $\theta_{3,2,4} = P(X_3 = 2 \mid Y = 4)$) and π for the class priors (e.g., $\pi_3 = P(Y = 3)$) similar to the lecture slides.)

b. [4 points] If the features are not independent conditioned on the label, one has to estimate the entire joint distribution $P(X_1, X_2, X_3, X_4 \mid Y = i)$, for all $i \in \{1, 2, 3, 4\}$. Determine the number of free parameters that one has to estimate in such a scenario.

c. [1 point] Based on the numbers of parameters you found in parts (a) and (b), explain one advantage of assuming conditional independence.

2 Naive Bayes in Practice

In this problem we will use the naïve Bayes algorithm to classify movie reviews as positive, neutral or negative. A simple approach involves maintaining a vocabulary of words that commonly occur in movie reviews and using the frequency of their occurrence in the three classes to classify movie reviews.

We are given the vocabulary $V = \{1 : \text{“incredible”}, 2 : \text{“plot”}, 3 : \text{“great”}, 4 : \text{“amazing”}, 5 : \text{“okay”}, 6 : \text{“decent”}, 7 : \text{“movie”}, 8 : \text{“no”}, 9 : \text{“acting”}, 10 : \text{“waste”}\}$. We will use V_i to represent the ith word in V. The training set provided includes four positive reviews:

- “great movie amazing”
- “incredible movie”
- “great acting amazing plot”
- “amazing acting amazing plot”

two neutral reviews:

- “okay movie”
- “decent no amazing acting”

and two negative reviews:

- “amazing waste”
- “no movie plot”

Recall that the naïve Bayes classifier is a generative classifier, where the probability of an input $\mathbf{x} = [x_1, x_2, \cdots, x_n]^\top$ depends on its class y. In our case the input vector $\mathbf{x}$ corresponding to each movie review has $n = 10$ equal to the number of words in the vocabulary V, where each entry x_i is equal to the number of times word V_i occurs in $\mathbf{x}$.

a. [2 points] Calculate the naïve Bayes estimates of $\Pr(y = 1)$, $\Pr(y = 2)$ and $\Pr(y = 3)$ from the training data, where $y = 1$ corresponds to positive reviews, $y = 2$ to neutral reviews and $y = 3$ to negative reviews.

b. [1 point] List the feature vector $\mathbf{x}$ for each positive review in the training set.

c. [2 points] In the naïve Bayes model, the likelihood of a sentence with feature vector $\mathbf{x}$ given a class c is

$$\Pr(\mathbf{x} \mid y = c) = \prod_{k=1}^{n} (\theta_{c,k})^{x_k}$$

where $\theta_{c,k}$ is the weight of word k among all words of class c. Calculate the maximum likelihood estimate of $\theta_{1,4}$, $\theta_{1,7}$, $\theta_{2,4}$, $\theta_{2,7}$, $\theta_{3,4}$ and $\theta_{3,7}$.

d. [3 points] Given a new review “amazing movie”, decide whether it is positive, neutral or negative, based on the naïve Bayes classifier, learned from the above data.

e. [4 points] Use Laplacian smoothing with $\alpha = 1$ to decide whether the review “decent movie” is positive, neutral or negative. In one sentence, describe the problem we would encounter if we had not used Laplacian smoothing.

3 Logistic Regression in Practice

Given a training set $D = \left\{ \left(\mathbf{x}^{(1)}, y^{(1)}\right), \ldots, \left(\mathbf{x}^{(N)}, y^{(N)}\right) \right\}$ where $\left(\mathbf{x}^{(i)} \in \mathbb{R}^d, y^{(i)} \in \{0,1\}\right)$ is the feature vector and the binary label for data point i, we want to find the parameters $\hat{\mathbf{w}}$ that maximize the likelihood for the training set, assuming a parametric model of the form

$$p(y = 1 \mid \mathbf{x}, \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})}.$$

The conditional log likelihood of the training set is

$$L(\mathbf{w}) = \sum_{i=1}^{N} \left[y^{(i)} \log p\left(y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w}\right) + \left(1 - y^{(i)}\right) \log\left(1 - p\left(y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w}\right)\right) \right],$$

and the gradient is

$$\nabla L(\mathbf{w}) = \sum_{i=1}^{N} \left(y^{(i)} - p\left(y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w}\right)\right) \mathbf{x}^{(i)}. \quad (2)$$

a. [2 points] Is it possible to get a closed form for the parameters $\hat{\mathbf{w}}$ that maximize the conditional log likelihood? If it is possible, give the closed-form solution. If not, how would you compute $\hat{\mathbf{w}}$ in practice? Explain your method to find $\hat{\mathbf{w}}$ in a few sentences or provide a short pseudo-code.

For a binary logistic regression model, we predict $y = 1$ when $p(y = 1 \mid \mathbf{x}) \geq 0.5$. Assume that the decision boundary occurs when $P(y = 1 \mid \mathbf{x}, \mathbf{w}) = P(y = 0 \mid \mathbf{x}, \mathbf{w})$.

b. [4 points] Find the decision boundary, which is the set of $\mathbf{x}$ satisfying $P(y = 1 \mid \mathbf{x}, \mathbf{w}) = P(y = 0 \mid \mathbf{x}, \mathbf{w})$.

c. [1 point] Is this model a linear classifier?

d. [2 points] Now, let us assume that in our application of this model, our tolerance of mis-classifying samples with the true label $y = 0$ is much lower than our tolerance to mis-classifying samples with true label $y = 1$ (i.e., we do not want to estimate a data point as belonging to class 0 if its true class is 1. However, we can tolerate the error of estimating a data point belong to class 1 if its true class is 0.) For example, in predicting the presence of a disease, we may be much more tolerant of mis-classifying a healthy person (true label $y = 0$) as having the disease ($y = 1$) compared to mis-classifying a person with the disease (true label $y = 1$) as not having it ($y = 0$). Now suppose you have trained a logistic regression model to obtain the model weights $\mathbf{w}$. How would you adjust the prediction with weights $\mathbf{w}$ to decrease the mis-classification of samples with true label $y = 1$?

4 Solving Logistic Regression

The cross-entropy loss function for a logistic regression task on a dataset $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ can be written as

$$L(\mathbf{w}) = -\sum_{i=1}^{n} \left[y_i \log p(y_i \mid \mathbf{x}_i, \mathbf{w}) + (1 - y_i) \log(1 - p(y_i \mid \mathbf{x}_i, \mathbf{w}))\right],$$

In this question, we will show that finding the MLE (maximum likelihood estimator) is equivalent to minimizing the cross-entropy loss in Eq. (3). However, unlike linear regression, it is difficult to derive a closed-form solution for logistic regression.

a. [3*points*] Starting from the definition of negative log-likelihood (Eq. (4) below), show that it is equal to the cross-entropy loss (Eq. (3)).

The negative log-likelihood is given by:

$$-\ell(\mathbf{w}) = -\log\left(\prod_{i=1}^{n}\left(\frac{1}{1+\exp(-\mathbf{w}^\top \mathbf{x}_i)}\right)^{y_i}\left(1 - \frac{1}{1+\exp(-\mathbf{w}^\top \mathbf{x}_i)}\right)^{1-y_i}\right). \quad (4)$$

Show that this is equivalent to the cross-entropy loss:

$$L(\mathbf{w}) = -\sum_{i=1}^{n}\left[y_i \log\left(\frac{1}{1+\exp(-\mathbf{w}^\top \mathbf{x}_i)}\right) + (1-y_i)\log\left(1 - \frac{1}{1+\exp(-\mathbf{w}^\top \mathbf{x}_i)}\right)\right]. \quad (3)$$

b. [4 points] Since it is difficult to directly optimize Eq. (3), gradient descent algorithms are usually used to find the optimum, as is discussed in the lecture. Show that the negative log-likelihood is a convex function. You may use the facts that the sum of convex functions is also convex, and that if f and g are both convex, twice differentiable and g is non-decreasing, then $g(f)$ is convex.
Hint: A function f is convex if for any $x_1, x_2 \in \text{Domain}(f)$,

$$f(tx_1 + (1-t)x_2) \le tf(x_1) + (1-t)f(x_2), \quad \forall t \in [0,1] \quad (5)$$

Also, a twice-differentiable function is convex if and only if the Hessian is positive semi-definite.

We now turn our attention to another algorithm known as iterative weighted least squares, which is based on the Newton-Raphson algorithm. Let $\mathbf{w}^{(k)}$ denote the parameter vector $\mathbf{w}$ at the kth iteration. Then, the update rule of iterative weighted least-squares is as follows:

$$\mathbf{w}_{(k+1)} = \mathbf{w}_{(k)} - \left(\nabla^2 \mathcal{L}\mathbf{w}_{(k)}\right)^{-1} \nabla L(\mathbf{w}_{(k)}),$$

where $\nabla L(\mathbf{w})$ and $\nabla^2 L(\mathbf{w})$ are the gradient and Hessian of the negative log-likelihood in Eq. (3), respectively. Consider the following notations:

- $\mathbf{y}$ is an $N \times 1$ column vector with the i-th element being the label y_i,
- $\mathbf{X}$ is an $N \times d$ matrix with $\mathbf{x}_i^\top$ being the i-th row,
- $\mathbf{W}_k$ is an $N \times N$ diagonal matrix with the i-th diagonal element being $\frac{\exp(-\mathbf{w}_k^\top \mathbf{x}_i)}{(1+\exp(-\mathbf{w}_k^\top \mathbf{x}_i))^2}$,
- $\mathbf{p}_k$ is an $N \times 1$ column vector with the i-th element being $\frac{1}{1+\exp(-\mathbf{w}_k^\top \mathbf{x}_i)}$,
- $\mathbf{z}_k = \mathbf{X}\mathbf{w}_k + \mathbf{W}_k^{-1}(\mathbf{y} - \mathbf{p}_k)$.

It can be shown that:

- $\nabla L(\mathbf{w}_k) = -\mathbf{X}^\top(\mathbf{y} - \mathbf{p}_k)$,
- $\nabla^2 L(\mathbf{w}_k) = \mathbf{X}^\top \mathbf{W}_k \mathbf{X}$.

c [5 points] Using the above, prove that

$$\mathbf{w}_{k+1} = (\mathbf{X}^\top \mathbf{W}_k \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{W}_k \mathbf{z}_k.$$

Note: The name of the algorithm comes from the observation that the algorithm at each step is solving the weighted least squares problem,

$$\mathbf{w}_{k+1} = \arg\min_{\mathbf{w}_k} (\mathbf{z}_k - \mathbf{X}\mathbf{w}_k)^\top \mathbf{W}_k (\mathbf{z}_k - \mathbf{X}\mathbf{w}_k).$$