ML-HW2: Mathematical Derivation

313553058
Ian Tsai

GitLink: https://github.com/IaTsai/ML-HW2

HW2-1: Naive Bayes Classifier

1. Mathematical Model

Naive Bayes models the class probability:

$$P(Y|X) \propto P(Y) \prod_i P(X_i|Y) \tag{1}$$

- $P(Y)$ is the prior (estimated from training samples).
- $P(X_i|Y)$ is the likelihood (discrete/continuous).
- All probabilities are computed in log-space to avoid underflow.

2.1 Discrete Mode Implementation

Implemented Code:

```
image = np.zeros((10, image_size, 32), dtype=np.int32)
...
image[label][j][grayscale // 8] += 1
...
if likelihood == 0:
    likelihood = np.min(image[j][k][np.nonzero(image[j][k])])
...
prob[j] += np.log(prior[j] / train_size)
prob[j] += np.log(likelihood / image_sum[j][k])
```

Explanation:

- Pixel grayscale is mapped to bin index, each pixel value (0-255) into 32 bins (each bin represents 8 levels).
- Count pixel frequencies for each class from training data.
- Frequencies stored in a 3D array `[class][pixel][bin]`.
- **Debug:** If the bin count is zero, use smoothing to avoid crashing the log by substituting a minimal non-zero value.
- Classifier sums the log prior and log-likelihoods over all pixels.

2.2 Continuous Mode Implementation

Implemented Code:

```
mean[label][j] += grayscale
mean_square[label][j] += grayscale ** 2
...
mean[i][j] /= prior[i]
mean_square[i][j] /= prior[i]
var[i][j] = mean_square[i][j] - mean[i][j] ** 2
var[i][j] = 1000 if var[i][j] == 0 else var[i][j]
...
likelihood = -0.5 * (np.log(2 * math.pi * var[j][k]) +
                ((test_image[k] - mean[j][k]) ** 2) / var[j][k])
prob[j] += np.log(prior[j] / train_size)
prob[j] += likelihood
```

Explanation:

- MLE estimates μ and σ^2 from training data.
- Gaussian PDF (in log form) computes $P(X_i|Y)$.
- Sum all pixel-wise log-likelihoods and add the class prior; the class with the highest score is the predicted result.

Log-Likelihood under Gaussian Naive Bayes

Example: Given a single test sample (flattened pixel array), and estimated Gaussian parameters (mean & variance) for each class:

- For each class j, sum the log prior and the pixel-wise log-likelihoods:

$$\log P(Y = j) + \sum_i \log P(X_i \mid Y = j) \tag{2}$$

Code illustration:

```
prob[j] += np.log(prior[j] / train_size)
for k in range(image_size):
    likelihood = -0.5 * (np.log(2 * math.pi * var[j][k]) +
                    ((test_image[k] - mean[j][k]) ** 2) / var[j][k])
    prob[j] += likelihood
```

This additive formulation across pixels accounts for the overall class score. The class with the highest (least negative) final `prob[j]` becomes the prediction.

4. Visualization and Output

```
print('Posterior (in log scale):')
print(f'Prediction: {pred}, Ans: {answer}\n')
DrawImagination(...)
```

5. Execution Interface

- `python 2-1_naive_bayes_classifier.py --train`: Trains Discrete or Continuous mode.
- `python 2-1_naive_bayes_classifier.py`: Loads saved results and displays imaginations.

HW2-2: Beta-Binomial Bayesian Online Learning

Note: In accordance with the assignment's requirements, no library-based sampling (e.g., `numpy.random.beta`) is used. All distributions are computed analytically, and likelihoods are implemented manually.

1.1 Mathematical Proof

Given:

- Prior: $\text{Beta}(a, b)$
- Data: m successes (1s), n failures (0s)

Posterior Update:

$$P(p|D) \propto p^{a+m-1}(1-p)^{b+n-1} \tag{3}$$

$$a' = a + m, \quad b' = b + n \tag{4}$$

This posterior update can be derived by applying Bayes' theorem:

$$P(p|D) \propto P(D|p) \cdot P(p) \tag{5}$$

Assuming:

- Binomial likelihood: $P(D|p) = p^m(1-p)^n$
- Beta prior: $P(p) = \frac{1}{B(a,b)}p^{a-1}(1-p)^{b-1}$

Then:

$$P(p|D) \propto p^m(1-p)^n \cdot p^{a-1}(1-p)^{b-1} = p^{a+m-1}(1-p)^{b+n-1} \tag{6}$$

Therefore, the posterior is also a Beta distribution: $\text{Beta}(a + m, b + n)$

Case Example

Example 1: Uniform Prior (a = 0, b = 0)

Input:

```
input a: 0
input b: 0
case 1: 0101010101001011010101
```

Step-by-step:

- $m = 11$ (number of '1's)
- $n = 11$ (number of '0's)

Posterior update:

$$a' = a + m = 0 + 11 = 11, \quad b' = b + n = 0 + 11 = 11 \tag{7}$$

Output:

```
Likelihood: 0.16818809509277344
Beta prior:     a = 0  b = 0
Beta posterior: a = 11 b = 11
```

Example 2: Informative Prior (a = 10, b = 1)

Input:

```
input a: 10
input b: 1
case 1: 0101010101001011010101
```

Step-by-step:

- $m = 11$
- $n = 11$

Posterior update:

$$a' = a + m = 10 + 11 = 21, \quad b' = b + n = 1 + 11 = 12 \tag{8}$$

Output:

```
Likelihood: 0.16818809509277344
Beta prior:     a = 10 b = 1
Beta posterior: a = 21 b = 12
```

Code Illustration

```
import sys, math

a = int(input("input a: "))
b = int(input("input b: "))

with open(sys.argv[1]) as f:
    for line in f:
        line = line.strip()
        count1 = line.count('1')
        count0 = line.count('0')

        likelihood = (math.gamma(a + b) / (math.gamma(a) * math.gamma(b
            ))) * \
                     (math.gamma(a + count1) * math.gamma(b + count0) /
                      math.gamma(a + b + count1 + count0))

        print("Likelihood:", likelihood)
        print(f"Beta prior:     a = {a}  b = {b}")

        a += count1
        b += count0

        print(f"Beta posterior: a = {a}  b = {b}\n")
```

Note: This Gamma-function-based implementation of the likelihood follows the theoretical Beta-Binomial formulation. However, the actual implementation in `2-2_online_learning.py` computes the likelihood using the standard Binomial model:

$$\text{Likelihood} = C(N, m) \cdot P^m \cdot (1 - P)^{N-m} \tag{9}$$

where $P = m/N$, and $C(N, m)$ is the binomial coefficient.

3. Output Format

```
Likelihood: ...
Beta prior:     a = X  b = Y
Beta posterior: a = X' b = Y'
```

HW2-3: Mathematical Derivation

This section provides complete mathematical proofs for two conjugate prior relationships required in the assignment:

1. Beta-Binomial Conjugation
2. Gamma-Poisson Conjugation

Proof 1: Beta-Binomial Conjugation

Objective: Show that the Beta distribution acts as a conjugate prior to the Binomial likelihood, including deriving the posterior distribution.

Problem Setup

- **Likelihood:** Binomial distribution

$$P(D|p) = \binom{N}{m} p^m (1-p)^n \tag{10}$$

  where $N = m + n$ (total trials), m = successes, n = failures

- **Prior:** Beta distribution

$$P(p) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} p^{a-1}(1-p)^{b-1} = \frac{1}{B(a,b)} p^{a-1}(1-p)^{b-1} \tag{11}$$

- **Goal:** Prove that the posterior is also a Beta distribution

Derivation

Given:

- Prior: Beta(a, b)
- Data: m successes (1s), n failures (0s)
- Total trials: $N = m + n$

Step 1: Write out Posterior (Bayes' Theorem)

$$P(p|D) \propto P(D|p) \cdot P(p) \tag{12}$$

Step 2: Binomial Likelihood

$$P(D|p) = \binom{N}{m} p^m (1-p)^n \tag{13}$$
$$\propto p^m (1-p)^n \tag{14}$$

(The binomial coefficient $\binom{N}{m}$ is a constant and can be ignored)

Step 3: Beta Prior

$$P(p) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} p^{a-1}(1-p)^{b-1} \tag{15}$$
$$\propto p^{a-1}(1-p)^{b-1} \tag{16}$$

(The normalization constant can be ignored)

Step 4: Calculate Posterior

$$\begin{aligned} P(p|D) &\propto P(D|p) \cdot P(p) && (17) \\ &\propto [p^m(1-p)^n] \cdot \left[p^{a-1}(1-p)^{b-1}\right] && (18) \\ &= p^{m+a-1}(1-p)^{n+b-1} && (19) \end{aligned}$$

Step 5: Identify Posterior Distribution

$$P(p|D) \propto p^{a'-1}(1-p)^{b'-1} \tag{20}$$

where:

$$\begin{aligned} a' &= a + m && (21) \\ b' &= b + n && (22) \end{aligned}$$

Conclusion

The Beta distribution is a **conjugate prior** for the Binomial likelihood. The parameter update rules are:

- $a' = a + m$ (shape parameter increases by number of successes)
- $b' = b + n$ (shape parameter increases by number of failures)

Therefore:

$$\boxed{\text{Posterior} = \text{Beta}(a + m, b + n)} \tag{23}$$

This is why in online learning (HW2-2), we can directly update Beta parameters without recomputing the entire distribution.

Proof 2: Gamma-Poisson Conjugation

Objective: Show that the Gamma distribution acts as a conjugate prior to the Poisson likelihood, including deriving the posterior distribution.

Problem Setup

- **Likelihood:** Poisson distribution
$$P(X|\lambda) = \frac{\lambda^x}{x!}e^{-\lambda} \tag{24}$$
- **Prior:** Gamma distribution
$$P(\lambda) = \frac{\beta^\alpha}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\beta\lambda} \tag{25}$$
- **Goal:** Prove that the posterior is also a Gamma distribution

Derivation

Given:

- Data: Observed n events with total count $X = \sum_{i=1}^n x_i$
- Prior: Gamma(α, β)

Step 1: Write out Posterior (Bayes' Theorem)

$$P(\lambda|X) \propto P(X|\lambda) \cdot P(\lambda) \tag{26}$$

Step 2: Poisson Likelihood (n independent observations)

$$P(X|\lambda) = \prod_{i=1}^n \left[\frac{\lambda^{x_i}}{x_i!}e^{-\lambda}\right] \tag{27}$$
$$= \left[\prod_{i=1}^n \frac{1}{x_i!}\right] \lambda^{\sum x_i} e^{-n\lambda} \tag{28}$$
$$\propto \lambda^X e^{-n\lambda} \tag{29}$$

(The factorial terms are constants and can be ignored)

Step 3: Gamma Prior

$$P(\lambda) = \frac{\beta^\alpha}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\beta\lambda} \tag{30}$$
$$\propto \lambda^{\alpha-1}e^{-\beta\lambda} \tag{31}$$

(The normalization constant can be ignored)

Step 4: Calculate Posterior

$$P(\lambda|X) \propto P(X|\lambda) \cdot P(\lambda) \tag{32}$$
$$\propto \left[\lambda^X e^{-n\lambda}\right] \cdot \left[\lambda^{\alpha-1}e^{-\beta\lambda}\right] \tag{33}$$
$$= \lambda^{X+\alpha-1}e^{-(n+\beta)\lambda} \tag{34}$$

Step 5: Identify Posterior Distribution

$$P(\lambda|X) \propto \lambda^{\alpha'-1}e^{-\beta'\lambda} \tag{35}$$

where:

$$\alpha' = \alpha + X = \alpha + \sum_{i=1}^n x_i \tag{36}$$
$$\beta' = \beta + n \tag{37}$$

Conclusion

The Gamma distribution is a **conjugate prior** for the Poisson likelihood. The parameter update rules are:

- $\alpha' = \alpha + \sum x_i$ (shape parameter increases by total observed events)
- $\beta' = \beta + n$ (rate parameter increases by number of observations)

Therefore:

$$\boxed{\text{Posterior} = \text{Gamma}(\alpha + \sum x_i, \beta + n)} \tag{38}$$

Applications

Gamma-Poisson conjugation is commonly used in:

- Event count analysis (e.g., website visits per hour)
- Failure rate estimation (e.g., equipment failures per day)
- Text analysis (e.g., word frequency per document)