10.1. Ước lượng hợp lý tối đa (*Maximum Likelihood Function - MLE*)

Trong thống kê và học máy thì dữ liệu thường được diễn tả thông qua những phân phối xác suất. Phân phối xác suất thường là một hàm số được đặc trưng bởi những tham số nhất định. Đối với phân phối chuẩn tham số đặc trưng chính là cặp trung bình và phương sai $\{\mu, \sigma^2\}$. Đối với phân phối Poisson thì tham số đặc trưng là λ. Nếu đã biết về dạng hàm phân phối, làm thế nào để tìm ra các tham số phân phối hợp lý nhất cho một bộ dữ liệu? Đó chính là mục tiêu mà *ước lượng hợp lý tối đa* (*Maximum Likelihood Estimation*) ,viết tắt là *MLE*, sẽ giải quyết.

Trong thống kê *ước lượng hợp lý tối đa* là một phương pháp giúp ước lượng tham số phân phối của dữ liệu thông qua tối đa hoá *hàm hợp lý* sao cho dưới giả định của mô hình thống kê thì dữ liệu trở nên phù hợp nhất. Tính phù hợp được đo lường thông qua một hàm số được gọi là *hàm hợp lý* (*Likelihood Function*).

Giả định một bộ dữ liệu gồm N quan sát đầu vào là $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ được mô phỏng bởi một phân phối lý thuyết $f(.)$ sao cho phân phối lý thuyết được đặc trưng bởi một véc tơ tham số$\mathbf{w} = (w_0, w_1, \ldots, w_k)^\intercal$. Tập hợp tất cả những giá trị có thể của $\mathbf{w}$ được gọi là *không gian tham số* (*parameter space*) $\mathcal{W}$. Mục tiêu của *ước lượng hợp lý tối đa* là tìm kiếm véc tơ tham số $\mathbf{w}$ trong không gian tham số $\mathcal{W}$ sao cho giá trị *hàm hợp lý* là lớn nhất. Lớn nhất có nghĩa là phù hợp nhất. Thông thường *Hàm hợp lý* được ký hiệu là $L(\mathbf{w})$ là một hàm đối với $\mathbf{w}$, hàm số này đo lường xác suất đồng thời của tất cả các quan sát thuộc tập dữ liệu đầu vào $\mathcal{D}$.

$$L(\mathbf{w}) = P(\mathcal{D}|\mathbf{w})$$

Trong phương pháp *ước lượng hợp lý tối đa* thì $\mathcal{D}$ được xem như kết quả đã biết trước. Tìm kiếm được một véc tơ tham số $\hat{\mathbf{w}}$ phù hợp nhất cũng giống như đi tìm nguyên nhân để giải thích tốt nhất cho kết quả đã biết. Trong trường hợp các quan sát ngẫu nhiên có phân phối *độc lập và xác định* (*independent and identically distributed*) viết tắt là `iid`, thì *hàm hợp lý* sẽ bằng tích xác suất trên từng quan sát:

$$L(\mathbf{w}) = P(\mathcal{D}|\mathbf{w}) = P(\mathbf{x}_1, \ldots, \mathbf{x}_N|\mathbf{w}) = \prod_{i=1}^{N} P(\mathbf{x}_i|\mathbf{w})$$

véc tơ tham số $\mathbf{w}$ phù hợp nhất là nghiệm của bài toán tối ưu *hàm hợp lý*.

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} L(\mathbf{w})$$

Giải bài toán tối ưu của tích là không dễ dàng. Do đó chúng ta thường sử dụng logarith để chuyển từ tối ưu hàm hợp lý sang tối ưu log của hàm hợp lý (*log likelihood*). Để phân biệt với hàm hợp lý thì hàm *log của hàm hợp lý* được ký hiệu là một chữ l viết thường.

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} l(\mathbf{w}) = \arg\max_{\mathbf{w}} \log L(\mathbf{w})$$

Nếu dữ liệu phân phối `iid` thì bài toán tối ưu sẽ đẹp hơn rất nhiều:

$$\begin{aligned}\hat{\mathbf{w}} &= \arg\max_{\mathbf{w}} \log \prod_{i=1}^{N} P(\mathbf{x}_i|\mathbf{w}) \\ &= \arg\max \sum_{i=1}^{N} \log P(\mathbf{x}_i|\mathbf{w})\end{aligned}$$

Từ phương pháp ước lượng hợp lý tối đa chúng ta có thể chứng minh được ước lượng tham số của rất nhiều các phân phối khác nhau.

Thật vậy, chắc hẳn trong thống kê các bạn đã từng làm các dạng bài tập về ước lượng trung bình và phương sai của tổng thể dựa vào trung bình mẫu và kích thước mẫu. Ta có thể khái quát bài toán này thành một ví dụ như sau:

Bài tập:

Để ước lượng cân nặng trung bình của một người trưởng thành là một điều rất khó. Chúng ta không thể tìm ra con số chính xác về cân nặng trung bình của tất cả mọi người trưởng thành trên thế giới vì cân nặng luôn biến động và thực hiện quá trình này là tốn kém. Vì vậy chúng ta chỉ có thể tìm ra một ước lượng hợp lý nhất từ một mẫu nhỏ và lấy kết quả này đại diện cho tổng thể. Giả sử tiến hành đo mẫu gồm N người trưởng thành có cân nặng là $\mathcal{D} = \{x_1, x_2, \ldots, x_N\}$. Hãy ước lượng trung bình cân nặng của một người trưởng thành.

Lời giải:

Chúng ta giả định rằng cân nặng tuân theo phân phối chuẩn với trung bình là μ và phương sai σ^2. Như vậy theo phân phối chuẩn thì giá trị có xác suất xuất hiện cao nhất sẽ ở vị trí trung bình của phân phối. Tuy nhiên ta không chắc chắn rằng trung bình của N mẫu là ước lượng hợp lý nhất cho cân nặng. Chính vì thế chúng ta sử dụng phương pháp MLE để tìm ra ước lượng hợp lý tối đa. Bạn sẽ thấy ước lượng hợp lý nhất cho cân nặng chính là trung bình.

Thật vậy, vì cân nặng được giả định là tuân theo phân phối chuẩn nên xác suất tại một quan sát x_i sẽ được tính theo hàm mật độ xác suất:

$$P(x_i|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp - \frac{(x_i - \mu)^2}{2\sigma^2}$$

Contents

Cân nặng của mọi người được giả định là `iid` nên xác suất xảy ra của bộ dữ liệu được tính thông qua tích xác suất trên từng điểm dữ liệu. Khi đó các tham số μ, σ được ước lượng thông qua phương pháp MLE.

$$\begin{aligned}\hat{\mu}, \hat{\sigma} &= \arg\max_{\mu,\sigma} \sum_{i=1}^{N} \log P(x_i|\mu,\sigma) \\ &= \arg\max_{\mu,\sigma} \sum_{i=1}^{N} \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp - \frac{(x_i-\mu)^2}{2\sigma^2} \right] \\ &= \arg\max_{\mu,\sigma} [\underbrace{-\frac{N}{2}\log 2\pi}_{C} - N\log\sigma - \sum_{i=1}^{N} \frac{(x_i-\mu)^2}{2\sigma^2}] \\ &= \arg\max_{\mu,\sigma} [-N\log\sigma - \sum_{i=1}^{N} \frac{(x_i-\mu)^2}{2\sigma^2}] + C\end{aligned}$$

Đặt:

$$J(\mu,\sigma) \triangleq \arg\max_{\mu,\sigma} [-N\log\sigma - \sum_{i=1}^{N} \frac{(x_i-\mu)^2}{2\sigma^2}]$$

Điều kiện cần của cực trị theo đạo hàm bậc nhất:

$$\frac{\delta J(\mu,\sigma)}{\delta\mu} = -\sum_{i=1}^{N} \frac{(x_i-\mu)}{\sigma^2} = 0 \quad (1)$$

$$\frac{\delta J(\mu,\sigma)}{\delta\sigma} = -\frac{N}{\sigma} + \sum_{i=1}^{N} \frac{(x_i-\mu)^2}{\sigma^3} = 0 \quad (2)$$

Từ đẳng thức (1) ta suy ra:

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Đẳng thức (2) cho thấy:

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N} (x_i-\mu)^2$$

Đây chính là những ước lượng hợp lý nhất về trung bình và phương sai của mẫu. Từ những ước lượng này chúng ta có thể suy ra những ước lượng về khoảng tin cậy cho biến.

10.2. Ước lượng hậu nghiệm tối đa (*Maximum A Posteriori*)

Ở phương pháp *MLE* chúng ta ước lượng ra phân phối của dữ liệu dựa trên *hàm hợp lý*. *Hàm hợp lý* $P(\mathbf{x}_i|\mathbf{w})$ chỉ được tính trong điều kiện các tham số phân phối đã xác định. Điều đó có nghĩa rằng chúng ta không thể đưa thêm niềm tin của mình vào tham số để tác động lên xác suất. Đây là một hạn chế lớn, đặc biệt là trên những mô hình được hồi qui với kích thước mẫu nhỏ thì qui luật phân phối dựa trên tần suất không còn đáng tin cậy (hãy nhớ về ví dụ tung đồng xu). Khi đó kết quả dự báo sẽ chuẩn xác hơn nếu chúng ta đưa thêm niềm tin vào xác suất.

Đó chính là lý do mà *ước lượng hậu nghiệm tối đa* (*Maximum A Posteriori*), viết tắt là *MAP* ra đời, cho phép ta đưa thêm niềm tin về phân phối tham số vào mô hình. Về bản chất đây cũng là một phương pháp ước lượng **tham số** của một **phân phối xác suất**, nhưng khác biệt với *MLE* đó là thay vì tối đa hoá hàm *hợp lý* thì chúng ta tối đa hoá *xác suất hậu nghiệm*. Dựa vào công thức Bayes chúng ta có thể phân tích xác suất thành tích của hàm hợp lý với *xác suất tiên nghiệm* và điều chỉnh niềm tin vào mô hình thông qua *xác suất tiên nghiệm*. Bài toán tối ưu *MAP*:

$$\begin{aligned}\hat{\mathbf{w}} &= \arg\max_{\mathbf{w}} \ \log P(\mathbf{w}|\mathcal{D}) \\ &= \arg\max \ \log \frac{P(\mathcal{D}|\mathbf{w})P(\mathbf{w})}{P(\mathcal{D})} \\ &= \arg\max \ \underbrace{\log P(\mathcal{D}|\mathbf{w})}_{\text{log likelihood}} + \underbrace{\log P(\mathbf{w})}_{\text{prior}} - \underbrace{\log P(\mathcal{D})}_{\text{evidence}} \\ &= \arg\max \ \log P(\mathcal{D}|\mathbf{w}) + \log P(\mathbf{w})\end{aligned}$$

Dòng thứ nhất suy ra dòng thứ hai là do công thức Bayes. Dòng thứ 3 suy ra dòng thứ 4 là do xác suất $P(\mathcal{D})$ chỉ phụ thuộc vào dữ liệu mà không phụ thuộc vào $\mathbf{w}$. Do đó trong bài toán tối ưu đối với $\mathbf{w}$ ta có thể loại bỏ thành phần này.

Ta nhận thấy hàm mục tiêu trong phương pháp *MAP* có thêm *xác suất tiên nghiệm* (*prior*) so với *MLE*. Thành phần này cũng gần tương tự như thành phần *điều chuẩn* (*regularization term*) trong các mô hình hồi qui tuyến tính, hồi qui Logistic và SVM mà chúng ta đã học. Tác dụng của thành phần *điều chuẩn* đó là giảm thiểu hiện tượng *quá khớp* cho mô hình thông qua sự kiểm soát được áp đặt lên tham số của mô hình hồi qui.

Ưu điểm của phương pháp *MAP* đó là chúng ta có thể đưa thêm vào niềm tin của mình về mô hình thông qua xác suất $P(\mathbf{w})$ để tối đa hoá hàm mục tiêu. Điều này là rất quan trọng vì thông qua những lượt huấn luyện mô hình trên những bộ dữ liệu khác nhau thì chúng ta có thể suy ra được phân phối $P(\mathbf{w})$ một cách chắc chắn hơn và thông qua đó làm giảm hiện tượng *quá khớp*.

Trong trường hợp chúng ta xem phân phối của $\mathbf{w}$ là đồng nhất thì $\log P(\mathbf{w})$ là không đổi. Khi đó bài toán tối ưu *MAP* trở thành tối ưu *MLE*. Như vậy chúng ta có thể coi *MAP* là một bước phát triển mới, một phương pháp tổng quát hơn của *MLE* cho phép chúng ta thể hiện niềm tin của mình đối với mô hình.

10.3. Mô hình xác suất Naive Bayes

Mô hình xác suất Naive Bayes là mô hình mà xác suất dự báo được ước tính dựa trên công thức Naive Bayes.

Giả định bộ dữ liệu có biến đầu vào bao gồm d biến $\mathbf{x} = \{x_1, x_2, \ldots, x_d\}$. Những biến này được giả định là độc lập có điều kiện theo biến mục tiêu y. Trong đó biến mục tiêu y có giá trị nằm trong tập hợp nhãn $\mathcal{C} = \{1, 2, \ldots, C\}$. Giả định thêm rằng $\mathcal{H}$ là một giả thuyết về phân phối xác suất của biến đầu vào $\mathbf{x}$ tương ứng với từng giá trị của biến mục tiêu y. Khi đó phân phối của $\mathbf{x}$ sẽ phụ thuộc vào y và $\mathcal{H}$ nhưng phân phối của y không bị phụ thuộc vào $\mathcal{H}$. Một ước lượng điểm đối với *xác suất hậu nghiệm* theo công thức Bayes như sau:

$$
\begin{aligned}
P(y|\mathbf{x}, \mathcal{H}) &= P(y|x_1, x_2, \ldots, x_d, \mathcal{H}) \\
&= \frac{P(x_1, x_2, \ldots, x_d|y, \mathcal{H})P(y|\mathcal{H})}{P(x_1, x_2, \ldots, x_d|\mathcal{H})} \\
&= \frac{P(x_1, x_2, \ldots, x_d|y, \mathcal{H})P(y)}{P(\mathbf{x}|\mathcal{H})} \\
&= \frac{\underbrace{\prod_{i=1}^{d} P(x_i|y, \mathcal{H}))}_{\text{likelihood}} \underbrace{P(y)}_{\text{prior}}}{\underbrace{P(\mathbf{x}|\mathcal{H})}_{\text{evidence}}} \\
&\propto \prod_{i=1}^{d} P(x_i|y, \mathcal{H})P(y) \qquad (3)
\end{aligned}
$$

$P(y|\mathbf{x}, \mathcal{H})$ chính là ước lượng xác suất từ giả thuyết $\mathcal{H}$ sau khi đã biết $\mathbf{x}$. Xác suất này là mục tiêu mà chúng ta cần tối ưu. Điều đó cũng có nghĩa rằng nếu ground truth là $y = c$ thì mô hình *Naive Bayes* cần đưa ra dự báo cho khả năng xảy ra của nhãn c càng lớn càng tốt. Xác suất này sẽ được tính theo khai triển từ công thức Bayes như chúng ta thấy ở (3). Tiếp theo chúng ta cùng đi phân tích phép biến đổi *xác suất hậu nghiệm*.

Từ dòng 1 sang dòng 2 là do công thức Bayes. Mặt khác $\mathcal{H}$ là giả thuyết về phân phối của $\mathbf{x}$ nên giả thuyết này là độc lập với y. Điều này dẫn tới $P(y|\mathcal{H}) = P(y)$. Từ đó dòng 2 ta suy ra dòng 3. Tiếp theo các chiều dữ liệu đầu vào là độc lập có điều kiện theo y nên:

$$
P(x_1, x_2, \ldots, x_d|y, \mathcal{H}) = \prod_{i=1}^{d} P(x_i|y, \mathcal{H}) \qquad (4)
$$

Giả định (4) ở trên là một sự ngây ngô vì đối với những bộ dữ liệu lớn gồm nhiều chiều thì rất ít khi đạt được điều kiện lý tưởng về sự độc lập. Vì thế mô hình mới có tên gọi là *Naive Bayes* (tạm dịch là *Bayes ngây ngô*). Tuy nhiên thực nghiệm cho thấy giả định ngây ngô này lại khá hiệu quả trong nhiều lớp mô hình phân loại của học có giám sát mà chúng ta sẽ tìm hiểu về lý thuyết của chúng ở bài viết này.

Tiếp theo công thức ở dòng 4 là một công thức quen thuộc phân rã *xác suất hậu nghiệm* thành ba thành phần chính đó là likelihood, prior và evidence:

- *Likelihood*: Là phân phối xác suất của dữ liệu đầu vào $\mathbf{x}$ trong điều kiện đã biết biến mục tiêu y. Xác suất này thể hiện *tính phù hợp* (*goodness of fit*) của tham số phân phối được giả định trong giả thuyết $\mathcal{H}$. Đồng thời, *Likelihood* cũng cho biết mức độ đóng góp vào giải thích xác suất của y từ phía dữ liệu đầu vào $\mathbf{x}$. Xác suất này phụ thuộc vào tham số phân phối nên quá trình tối ưu theo Naive Bayes chủ yếu là dựa vào tìm kiếm bộ tham số sao cho *Likelihood* là tối đa.
- *Prior*: *Xác suất tiên nghiệm* của biến mục tiêu y. Chúng ta có thể đưa vào niềm tin của người làm mô hình vào khả năng xảy ra của y, thông qua đó tác động tới *xác suất hậu nghiệm* được dự báo. Thông thường đối với những bộ dữ liệu có kích thước lớn thì phân phối xác suất của y được ước lượng chính là tỷ lệ giữa các nhãn trong tập huấn luyện.
- *Evidence*: Là phân phối xác suất của dữ liệu không phụ thuộc vào giá trị của y. Với mỗi một giả thuyết $\mathcal{H}$ thì $P(\mathbf{x}|\mathcal{H})$ là cố định nên chúng ta suy ra *xác suất hậu nghiệm* sẽ đồng dạng với tích giữa *likelihood* và *xác suất tiên nghiệm*. Tức là chúng ta có thể bỏ qua *Evidence* trong quá trình tối ưu *xác suất hậu nghiệm*.

Chúng ta có một tính chất khá quan trọng về sự *chuẩn hoá xác suất* của *xác suất hậu nghiệm* trong công thức (3). Tức là tổng xác suất của toàn bộ các trường hợp sẽ bằng 1:

$$
\sum_{y} P(y|\mathbf{x}, \mathcal{H}) = \sum_{y} \frac{P(\mathbf{x}|y, \mathcal{H})P(y|\mathcal{H})}{P(\mathbf{x}|\mathcal{H})} = \sum_{y} \frac{P(\mathbf{x}|y, \mathcal{H})P(y|\mathcal{H})}{\sum_{y} P(\mathbf{x}|y, \mathcal{H})P(y|\mathcal{H})} = 1
$$

Điều đó cho thấy các ước lượng xác suất từ công thức Bayes của *xác suất hậu nghiệm* bản thân nó đã được chuẩn hoá để trở thành một phân phối xác suất. $P(y|\mathbf{x}, \mathcal{H})$ là xác suất đối với một khả năng của y nằm trong tập nhãn $\mathcal{C}$. Như vậy nhãn dự báo $\hat{y}$ phải là nhãn mà có khả năng xảy ra là lớn nhất. Điều đó có nghĩa rằng:

$$
\begin{aligned}
\hat{y} &= \arg\max_{y \in \mathcal{C}} P(y|\mathbf{x}, \mathcal{H}) \\
&= \arg\max_{y \in \mathcal{C}} \prod_{i=1}^{d} P(x_i|y, \mathcal{H})P(y)
\end{aligned}
$$

Như vậy mô hình *Naive Bayes* thực chất là ước lượng một *xác suất hậu nghiệm* nên chúng ta có thể dựa trên *MAP* để tìm ra các tham số phân phối cho dữ liệu.

Mô hình Naive Bayes dựa trên giả định khá *ngây ngô* về sự độc lập có điều kiện giữa các chiều dữ liệu nhưng giả định ngây ngô này lại cho thấy hoạt động hiểu quả trong nhiều bài toán, đặc biệt là các bài toán về phân loại tin rác và phân loại văn bản. Chính nhờ giả định *ngây ngô* mà bài toán tối ưu xác suất đã trở nên dễ dàng hơn nhờ xác suất được ước lượng trên từng chiều độc lập.

Chi phí huấn luyện cho bài toán Naive Bayes cũng ít tốn kém hơn so với các bài toán phân loại khác trong Machine Learning. Vì chúng ta không cần phải giải bài toán tối ưu trên dữ liệu nhiều chiều. Giả định *ngây ngô* đã phân rã xác suất về những chiều đơn lẻ và độc lập. Dẫn tới thực chất tối ưu *xác suất hậu nghiệm* là tối ưu phân phối xác suất trên từng chiều độc lập. Tuỳ thuộc vào dữ liệu là liên tục hoặc hạng mục mà ước lượng xác suất trên từng chiều sẽ được tính dựa trên hàm mật độ xác suất hoặc dựa trên tần suất. Nhưng nhìn chung thì những tối ưu này đều khá nhẹ nhàng.

10.3.1. Gaussian Naive Bayes

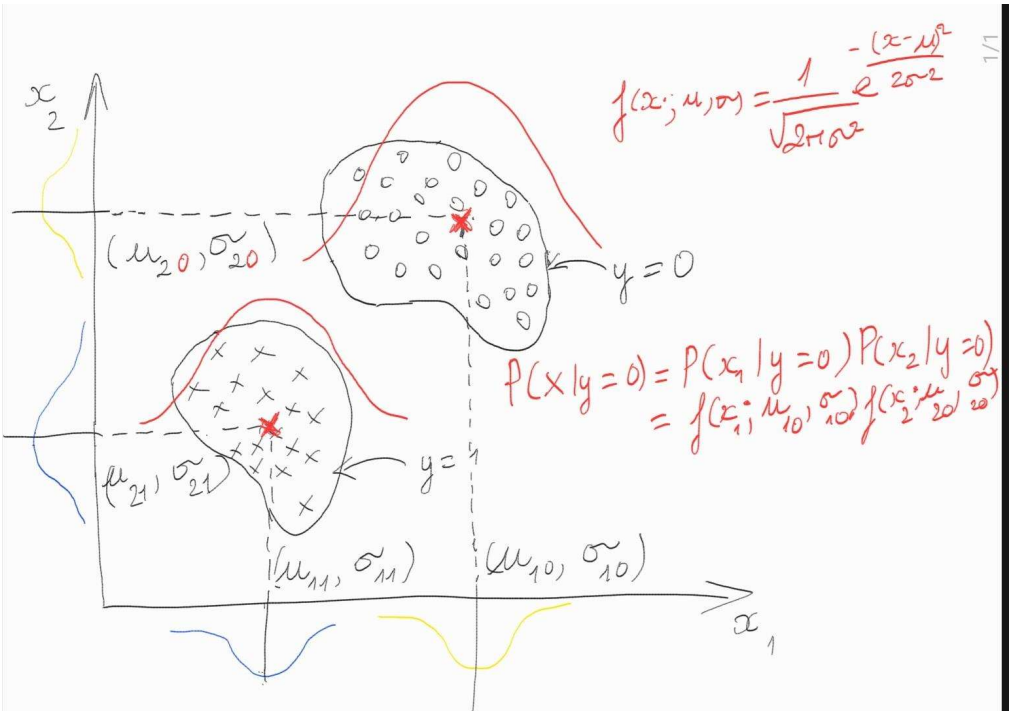
Trong mô hình Gaussian Naive Bayes, xác suất của một chiều dữ liệu x_i đối với một nhãn cụ thể $y = c$ được giả định dựa trên phân phối Gaussian và đặc trưng bởi hai tham số phân phối là trung bình μ_{ic} và phương sai σ_{ic}^2. Khi đó xác suất được ước lượng từ phân phối Gaussian:

$$P(x_i|y=c) = f(x_i; \mu_{ic}, \sigma_{ic}) = \frac{1}{\sqrt{2\pi\sigma_{ic}^2}} \exp\left(-\frac{(x_i - \mu_{ic})^2}{2\sigma_{ic}^2}\right)$$

Để ước lượng ra hai tham số μ_{ic} và σ_{ic} chúng ta sử dụng phương pháp ước lượng MLE trên toàn bộ dữ liệu. Tức là μ_{ic} và σ_{ic} phải là nghiệm của hàm *hợp lý*:

$$\hat{\mu}_{ic}, \hat{\sigma}_{ic} = \arg\max \prod_{j=1}^{N} P(x_i^{(j)}|y^{(j)} = c)$$

Trong đó (j) chính là chỉ số của quan sát thứ j trong bộ dữ liệu. Từ bài tập trong chương ước lượng hợp lý tối đa ta có thể dễ dàng suy ra giá trị ước lượng của hai tham số $\hat{\mu_{ic}}, \hat{\sigma_{ic}}$ tương ứng với trung bình và phương sai của các quan sát có nhãn là c.



Hình 1: Các điểm chấm tròn thuộc về nhãn $y = 0$ và dấu nhân thuộc về nhãn $y = 1$. Trên hình vẽ chúng ta thực hiện các phép chiếu lên các trục x_1 và x_2 để thu được phân phối trên từng trục. Đường màu xanh thể hiện phân phối đối với nhãn 1 và màu vàng là phân phối của nhãn 0. Phép chiếu từ các điểm có nhãn 1 lên trục x_1 ta thu được một phân phối chuẩn $\mathbf{N}(\mu_{11}, \sigma_{11})$ như hình vẽ. Như vậy theo phân phối chuẩn thì những điểm càng gần tâm của nhãn $y = 1$ thì xác suất $P(x_1|y = 1) = f(x_1; \mu_{11}, \sigma_{11})$ càng lớn. Như vậy về bản chất xác suất trên từng chiều dữ liệu chính là một thước đo mức độ tương đồng đến tâm của nhãn. Xác suất này càng lớn thì thì các điểm dữ liệu sẽ càng gần tâm của nhãn và do đó khả năng cao chúng được phân loại về nhãn là chính xác.

Thông thường mô hình Gaussian Naive Bayes sẽ áp dụng trên dữ liệu đầu vào là những biến liên tục. Để xây dựng mô hình `Gaussian Naive Bayes` thì trong sklearn thì chúng ta sử dụng class `sklearn.naive_bayes.GaussianNB`. Bên dưới chúng ta sẽ thực hành huấn luyện mô hình này trên bộ dữ liệu iris.

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
from sklearn.naive_bayes import GaussianNB

X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=0)

gnb = GaussianNB()
gnb.fit(X_train, y_train)

y_pred = gnb.predict(X_test)
print(classification_report(y_pred, y_test))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       1.00      0.90      0.95        21
           2       0.87      1.00      0.93        13

    accuracy                           0.96        50
   macro avg       0.96      0.97      0.96        50
weighted avg       0.97      0.96      0.96        50
```

Như vậy trên tập kiểm tra mô hình dự báo có độ chính xác trung bình trên cả ba loài hoa đạt 96%. Đây không phải là một độ chính xác quá cao. Trên thực tế thì mô hình `Gaussian Naive Bayes` thường không phải là một lớp mô hình mạnh trong những bài toán phân loại có dữ liệu đầu vào là những biến liên tục. Ưu điểm của `Gaussian Naive Bayes` đó là có chi phí huấn luyện thấp, tốc độ tính toán nhanh và hoạt động trực tiếp trên những bài toán phân loại đa lớp mà không cần phải chuyển sang những bài toán `one-vs-one` hoặc `one-vs-rest`.

10.3.2. Multinomial Naive Bayes

Đây là phương pháp thường được sử dụng trong bài toán phân loại văn bản và thực nghiệm cho thấy là một phương pháp khá hiệu quả. Đầu tiên, chúng ta sẽ xây dựng một từ điển bao gồm toàn bộ các từ xuất hiện trong toàn bộ các văn bản. Gia sử từ điển này là tập $\mathcal{D} = \{x_1, x_2, \ldots, x_d\}$, trong đó x_i là một từ ở vị trí thứ i trong từ điển. Từ điển $\mathcal{D}$ luôn có kích thước cố định là d. Thông qua $\mathcal{D}$, một văn bản $\mathbf{x}_j$ bất kì được đặc trưng bởi một véc tơ tần suất $(N_{1j}, N_{2j}, \ldots, N_{dj})$ có độ dài bằng độ dài từ điển. Trong đó N_{ij} đại diện cho tần suất của từ x_i trong từ điển xuất hiện trong văn bản $\mathbf{x}_j$. Xác suất để văn bản $\mathbf{x}_j$ rơi vào lớp $y = c$ được tính theo công thức xác suất Bayes:

$$\begin{aligned} P(y = c|\mathbf{x}_j) &= \frac{P(\mathbf{x}_j|y = c)P(y = c)}{P(\mathbf{x}_j)} \\ &\propto \underbrace{P(y = c)}_{\text{prior}} \underbrace{\prod_{i=1}^{d} P(x_i|y = c)^{N_{ij}}}_{\text{likelihood}} \quad (5) \end{aligned}$$

Xác suất tiên nghiệm (*prior*) được tính toán khá dễ dàng dựa trên thống kê tỷ lệ quan sát rơi vào từng lớp văn bản.

likelihood thực chất là một phân phối multinomial về khả năng xuất hiện đồng thời các từ trong văn bản $\mathbf{x}_j$ với tần suất $(N_{1j}, N_{2j}, \ldots, N_{dj})$. Để tính được *likelihood* thì chúng ta phải tính được xác suất xuất hiện của từng từ trong một lớp văn bản có nhãn $y = c$. Ta kí hiệu xác suất này là $\lambda_{ic} = P(x_i|y = c)$. Đồng thời kí hiệu $\mathcal{C}$ là tập hợp indice của các văn bản thuộc lớp $y = c$. Dễ dàng nhận thấy:

$$\lambda_{ic} = \frac{\sum_{j \in \mathcal{C}} N_{ij}}{N_c}$$

Trong đó N_c là toàn bộ các từ (tính cả lặp lại) xuất hiện trong các văn bản thuộc lớp $y = c$. Dễ nhận thấy:

$$N_c = \sum_{i=1, j \in \mathcal{C}}^{d} N_{ij}$$

Trong một số tình huống khi một từ không xuất hiện trong văn bản thì sẽ có $\lambda_{ic} = 0$. Khi đó xác suất dự báo ở vế trái của (5) sẽ bằng 0 bất kể các xác suất tương ứng với các từ còn lại xuất hiện trong văn bản có lớn như thế nào. Điều này dẫn tới đánh giá sai lệch về kết quả dự báo. Chính vì thể để khắc phục hiện tượng xác suất bị triệt tiêu về 0 do thiếu từ thì chúng ta sử dụng phương pháp *Laplace smoothing*:

$$\lambda_{ic} = \frac{\sum_{j \in \mathcal{C}} N_{ij} + \alpha}{N_c + \alpha d}$$

Hệ số α được lựa chọn là một số dương. Chúng ta nhân αd ở mẫu là để tổng $\sum_{i=1}^{d} \lambda_{ic} = 1$. Thông thường hệ số α được lựa chọn bằng 1.

Sau khi tính được xác suất của toàn bộ các từ trong bộ từ điển đối với nhãn c ta thu được phân phối $[\lambda_{1c}, \lambda_{2c}, \ldots, \lambda_{dc}]$. Khi đó ta tính được xác suất dự báo:

$$P(y = c|\mathbf{x}_j) \propto P(y = c) \prod_{i=1}^{d} \lambda_{ic}^{N_{ij}}$$

Cách tính xác suất theo `Multinomial Naive Bayes` là khá đơn giản bởi chúng ta hoàn toàn ước lượng xác suất dựa trên thống kê về tần suất. Trong sklearn chúng ta sử dụng module `sklearn.naive_bayes.MultinomialNB` để xây dựng mô hình `Multinomial Naive Bayes`. Tiếp theo chúng ta sẽ phân loại văn bản thông qua module này.

Bộ dữ liệu:

Bộ dữ liệu huấn luyện được trích lọc từ `fetch_20newsgroups`. `fetch_20newsgroups` bao gồm 20 chủ đề khác nhau. Tuy nhiên ở đây ta chỉ lấy ra 1183 văn bản thuộc hai chủ đề là *cơ đốc giáo* có nhãn `soc.religion.christian` và *đồ hoạ máy tính* có nhãn `comp.graphics`.

```
from sklearn.datasets import fetch_20newsgroups

# Download bộ dữ liệu phân loại văn bản gồm 2 chủ đề  tôn giáo: 'soc.religion.christian',
'comp.graphics'].
categories = ['soc.religion.christian', 'comp.graphics']
twenty_train = fetch_20newsgroups(subset='train', categories=categories, shuffle=True,
random_state=42)
print('Total document: {}'.format(len(twenty_train.data)))
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
/tmp/ipykernel_43713/1671840682.py in <module>
      3 # Download bộ dữ liệu phân loại văn bản gồm 2 chủ đề  tôn giáo: 'soc.religion.christian',
'comp.graphics'].
      4 categories = ['soc.religion.christian', 'comp.graphics']
----> 5 twenty_train = fetch_20newsgroups(subset='train', categories=categories, shuffle=True,
random_state=42)
      6 print('Total document: {}'.format(len(twenty_train.data)))

~/miniconda3/envs/deepai-book/lib/python3.9/site-packages/sklearn/datasets/_twenty_newsgroups.py
in fetch_20newsgroups(data_home, subset, categories, shuffle, random_state, remove,
download_if_missing, return_X_y)
    262         if download_if_missing:
    263             logger.info("Downloading 20news dataset. This may take a few minutes.")
--> 264             cache = _download_20newsgroups(
    265                 target_dir=twenty_home, cache_path=cache_path
    266             )

~/miniconda3/envs/deepai-book/lib/python3.9/site-packages/sklearn/datasets/_twenty_newsgroups.py
in _download_20newsgroups(target_dir, cache_path)
     72
     73     logger.info("Downloading dataset from %s (14 MB)", ARCHIVE.url)
---> 74     archive_path = _fetch_remote(ARCHIVE, dirname=target_dir)
     75
     76     logger.debug("Decompressing %s", archive_path)

~/miniconda3/envs/deepai-book/lib/python3.9/site-packages/sklearn/datasets/_base.py in
_fetch_remote(remote, dirname)
   1446
   1447     file_path = remote.filename if dirname is None else join(dirname, remote.filename)
-> 1448     urlretrieve(remote.url, file_path)
   1449     checksum = _sha256(file_path)
   1450     if remote.checksum != checksum:

~/miniconda3/envs/deepai-book/lib/python3.9/urllib/request.py in urlretrieve(url, filename,
reporthook, data)
    266
    267             while True:
--> 268                 block = fp.read(bs)
    269                 if not block:
    270                     break

~/miniconda3/envs/deepai-book/lib/python3.9/http/client.py in read(self, amt)
    460             # Amount is given, implement using readinto
    461             b = bytearray(amt)
--> 462             n = self.readinto(b)
    463             return memoryview(b)[:n].tobytes()
    464         else:

~/miniconda3/envs/deepai-book/lib/python3.9/http/client.py in readinto(self, b)
    504         # connection, and the user is reading more bytes than will be provided
    505         # (for example, reading in 1k chunks)
--> 506         n = self.fp.readinto(b)
    507         if not n and b:
    508             # Ideally, we would raise IncompleteRead if the content-length

~/miniconda3/envs/deepai-book/lib/python3.9/socket.py in readinto(self, b)
    702         while True:
    703             try:
--> 704                 return self._sock.recv_into(b)
    705             except timeout:
    706                 self._timeout_occurred = True

~/miniconda3/envs/deepai-book/lib/python3.9/ssl.py in recv_into(self, buffer, nbytes, flags)
   1239                   "non-zero flags not allowed in calls to recv_into() on %s" %
   1240                   self.__class__)
-> 1241             return self.read(nbytes, buffer)
   1242         else:
   1243             return super().recv_into(buffer, nbytes, flags)

~/miniconda3/envs/deepai-book/lib/python3.9/ssl.py in read(self, len, buffer)
   1097         try:
   1098             if buffer is not None:
-> 1099                 return self._sslobj.read(len, buffer)
   1100             else:
   1101                 return self._sslobj.read(len)

KeyboardInterrupt:
```

Nội dung của object `twenty_train` sẽ bao gồm dữ liệu là văn bản được chứa trong list `twenty_train.data` và nhãn được chứa trong list `twenty_train.target`.

```
for i in range(5):
  print('----------------------------------> \n')
  print('Content: {} ...'.format(twenty_train.data[i][:100]))
  print('Target label: {}'.format(twenty_train.target[i]))
```

Chúng ta không thể đưa trực tiếp dữ liệu là văn bản vào để huấn luyện mô hình mà cần phải mã hoá chúng dưới dạng véc tơ.

Trong sklearn để xử lý văn bản, mã hoá kí tự sang số (*tokenizing*) và lọc bỏ *từ dừng* (*stopwords*) chúng ta hoàn toàn có thể sử dụng `sklearn.feature_extraction.text.CountVectorizer`. Class này sẽ giúp xây dựng một từ điển và biến đổi một văn bản thành một véc tơ đặc trưng theo tần suất xuất hiện các từ trong từ điển.

```
from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(twenty_train.data)
X_train_counts.shape
```

Ma trận `X_train_counts` thu được là một ma trận tần suất có số dòng bằng số lượng văn bản và số cột bằng kích thước của từ điển. Mỗi một dòng là một véc tơ tần suất xuất hiện của các từ trong từ điển. Những tần suất này được sắp xếp theo thứ tự khớp với thứ tự của các từ mà nó thống kê trong từ điển.

Huấn luyện mô hình Multinomial Naive Bayes

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
import numpy as np

# Khởi tạo mô hình
mnb_clf = MultinomialNB()

# Cross-validation với số K-Fold = 5 và thực hiện 1 lần cross-validation
cv = RepeatedStratifiedKFold(n_splits=5, n_repeats=1, random_state=1)
scores = cross_val_score(mnb_clf, X_train_counts, twenty_train.target, scoring='accuracy', cv=cv,
n_jobs=-1)
print('Mean Accuracy: {:.03f}, Standard Deviation Accuracy: {:.03f}'.format(np.mean(scores),
np.std(scores)))
```

Độ chính xác đạt được 98.9% là rất cao, đồng thời độ biến động của độ chính xác chỉ 0.6% khi thực hiện cross-validation. Điều đó cho thấy mô hình `Multinomial Naive Bayes` khá hiệu quả trong tác vụ phân loại văn bản.

Dự báo cho một văn bản mới

Để dự báo cho một nội dung văn bản mới chúng ta cần phải trải qua hai bước:

- Mã hoá văn bản sang véc tơ tần suất.
- Dự báo trên véc tơ tần suất.

Tất cả những bước này được thực hiện khá dễ dàng trên sklearn.

```
# Mã hoá câu văn sang véc tơ tần suất
docs_new = ['God is my love', 'OpenGL on the GPU is fast']
X_new_counts = count_vect.transform(docs_new)

# Dự báo
mnb_clf.fit(X_train_counts, twenty_train.target)
predicted = mnb_clf.predict(X_new_counts)

for doc, category in zip(docs_new, predicted):
  print('%r => %s' % (doc, twenty_train.target_names[category]))
```

10.4. Tổng kết

Như vậy qua bài viết này chúng ta đã được tìm hiểu thêm về sự khác biệt giữa hai trường phái *tần suất* và *bayesian* trong suy diễn thống kê. Điểm khác biệt giữa hai trường phái này đó là *tần suất* cho rằng xác suất là cố định, bất biến và phụ thuộc vào dữ liệu trong khi *bayesian* tạo ra sự linh hoạt hơn cho xác suất bằng cách đưa thêm vào niềm tin của người dự báo vào xác suất.

Phương pháp ước lượng tham số phân phối của dữ liệu dựa trên tối đa hoá hàm *hợp lý* được gọi là *MLE*. *MLE* trên thực tế là một trường hợp đặc biệt của *MAP* nếu phân phối của tham số được cho là đồng nhất.

Trong ước lượng *MAP* chúng ta tìm cách tối đa hoá *xác suất hậu nghiệm* thông qua phân rã chúng thành *xác suất tiên nghiệm* và *likelihood*. Thành phần *xác suất tiên nghiệm* có ý nghĩa như một *thành phần điều chuẩn* (*regularization term*) giúp kiểm soát giá trị của tham số ước lượng, thông qua đó giúp giảm thiểu hiện tượng *quá khớp* cho mô hình.

Naive Bayes là mô hình phân loại mà xác suất dự báo được tính dựa trên công thức Bayes. Trong mô hình *Naive Bayes* chúng ta dựa trên một giả thuyết *ngây ngô* đó là các biến đầu vào là độc lập có điều kiện theo biến mục tiêu. Như vậy quá trình tối ưu *xác suất tiên nghiệm* trở nên đơn giản hơn rất nhiều thông qua tối ưu trên từng chiều đặc trưng. Đối với biến đầu vào liên tục chúng ta ước lượng *likelihood* theo phân phối Gaussian trong khi các bài toán mà biến đầu vào dạng văn bản hoặc thứ bậc thì phân phối Multinomial được sử dụng. Mô hình *Naive Bayes* có chi phí tính toán thấp và tỏ ra khá hiệu quả đối với lớp các bài toán liên quan tới phân loại văn bản.

10.5. Bài tập

1. Trường phái *Bayesian* khác với *tần suất* (*Frequentist*) trong suy diễn thống kê như thế nào?
2. Ước lượng hợp lý tối đa *MLE* sẽ tìm ra ước lượng của tham số phân phối dựa trên hàm mục tiêu là gì?
3. Phương pháp *MAP* có mục tiêu là tối đa hoá hàm mục tiêu là gì?
4. Ưu điểm của *MAP* so với *MLE* là gì?
5. Trong mô hình xác suất *Naive Bayes* giả định nào được đặt ra và được xem là *ngây ngô*?
6. Làm thế nào để ước lượng ra tham số μ, σ trong phân phối *Gaussian* cho các biến đầu vào trong mô hình *Gaussian Naive Bayes*.
7. Phân phối *Multinomial Naive Bayes* thường được sử dụng trên dữ liệu dạng như thế nào?
8. Dựa vào bộ dữ liệu `fetch_20newsgroups`, hãy xây dựng mô hình phân loại chủ đề theo `Naive Bayes` cho 4 nhóm chủ đề là `'alt.atheism', 'comp.graphics', 'sci.med', 'soc.religion.christian'`.

10.6. Tài liệu

https://scikit-learn.org/stable/modules/naive_bayes.html

https://see.stanford.edu/materials/aimlcs229/cs229-notes1.pdf

https://deeplearningtheory.com/PDLT.pdf

https://towardsdatascience.com/probability-concepts-explained-maximum-likelihood-estimation-c7b4342fdbb1

https://wiseodd.github.io/techblog/2017/01/01/mle-vs-map/

https://towardsdatascience.com/mle-map-and-bayesian-inference-3407b2d6d4d9

Previous
10. Bạn là *Tần suất* (*Frequentist*) hay *Bayesian*?

Next
11. Giới thiệu về feature engineering

By Pham Dinh Khanh