Logistic Regression

Cornell CS 4/5780 — Spring 2022

Where we left off: Gaussian Naive Bayes. Real-valued features, model as

$$\mathbf{P}(x_\alpha \mid y = c) = \frac{1}{2\pi\sigma_{\alpha c}^2}\exp\left(-\frac{(x_\alpha - \mu_{\alpha c})^2}{2\sigma_{\alpha c}^2}\right),$$

where our parameters are $\mu_{\alpha c}$ and $\sigma_{\alpha c}^2$ (as well as the π_c for the class priors).

Suppose we are using a simplified model where $\sigma_{\alpha c}^2$ is independent of the class c, i.e. $\sigma_{\alpha c}^2 = \sigma_\alpha^2$ for some new parameters σ_α^2. What is $\hat{\mathcal{P}}(y|\mathbf{x})$ for Gaussian Naive Bayes? How much can you simplify the expression?

Machine learning algorithms can be (roughly) categorized into two categories:

- *Generative* algorithms, that estimate $P(\mathbf{x}_i, y)$ (often they model $P(\mathbf{x}_i|y)$ and $P(y)$ separately).
- *Discriminative* algorithms, that model $P(y|\mathbf{x}_i)$

The Naive Bayes algorithm is *generative*. It models $P(\mathbf{x}_i|y)$ and makes explicit assumptions on its distribution (e.g. multinomial, categorical, Gaussian, ...). The parameters of this distributions are estimated with MLE or MAP.

Logistic Regression is often referred to as the *discriminative* counterpart of Naive Bayes. Here, we model only the conditional distribution $P(y|\mathbf{x}_i)$ and assume that it takes on exactly this form

$$P(y|\mathbf{x}_i) = \frac{1}{1 + \exp\left(-y(\mathbf{w}^T\mathbf{x}_i + b)\right)}.$$

We make little assumptions on $P(\mathbf{x}_i|y)$, e.g. it could be Gaussian or Multinomial. Ultimately it doesn't matter, because we estimate the vector $\mathbf{w}$ and b directly with MLE or MAP to maximize the conditional likelihood of $\Pi_i P(y_i|\mathbf{x}_i; \mathbf{w}, b)$. For a lot more details, you can read the excellent book chapter by Tom Mitchell linked on the website.

To simplify things a bit, in this lecture we'll suppose that we absort the parameter b into $\mathbf{w}$ by adding an additional constant dimension, also known as using homogeneous coordinate, similar to what we did for the Perceptron.

Maximum likelihood estimate (MLE)

In MLE we choose parameters that **maximize the conditional likelihood**. The conditional data likelihood $P(\mathbf{y} \mid X, \mathbf{w})$ is the probability of the observed values $\mathbf{y} \in \mathbb{R}^n$ in the training data conditioned on the feature values $\mathbf{x}_i$. Note that $X = [\mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$. We choose the paramters that maximize this function and we assume that the y_i's are independent given the input features $\mathbf{x}_i$ and $\mathbf{w}$. So,

$$P(\mathbf{y} \mid X, \mathbf{w}) = \prod_{i=1}^{n} P(y_i \mid \mathbf{x}_i, \mathbf{w}).$$

Now if we take the log, e obtain

$$\log\left(\prod_{i=1}^{n} P(y_i|\mathbf{x}_i, \mathbf{w})\right) = -\sum_{i=1}^{n} \log(1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i})$$

$$\begin{aligned}\hat{\mathbf{w}}_{MLE} &= \underset{\mathbf{w}}{\operatorname{argmax}} -\sum_{i=1}^{n} \log(1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i}) \\ &= \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{n} \log(1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i})\end{aligned}$$

We need to estimate the parameters $\mathbf{w}$. To find the values of the parameters at minimum, we can try to find solutions for $\nabla_{\mathbf{w}} \sum_{i=1}^{n} \log(1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i}) = 0$. This equation has no closed form solution, so we will use Gradient Descent on the *negative log likelihood* $\ell(\mathbf{w}) = \sum_{i=1}^{n} \log(1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i})$.

Maximum a Posteriori (MAP) Estimate

In the MAP estimate we treat $\mathbf{w}$ as a random variable and can specify a prior belief distribution over it. We may use: $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$. This is the Gaussian approximation for LR.

Our goal in MAP is to find the *most likely* model parameters *given the data*, i.e., the parameters that **maximaize the posterior**.

$$P(\mathbf{w} \mid D) = P(\mathbf{w} \mid X, \mathbf{y}) \propto P(\mathbf{y} \mid X, \mathbf{w})\ P(\mathbf{w})$$

$$\hat{\mathbf{w}}_{MAP} = \underset{\mathbf{w}}{\operatorname{argmax}} \log\ (P(\mathbf{y} \mid X, \mathbf{w}) P(\mathbf{w})) = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{n} \log(1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i}) + \lambda \mathbf{w}^\top \mathbf{w},'$$

where $\lambda = \frac{1}{2\sigma^2}$. Once again, this function has no closed form solution, but we can use Gradient Descent on the *negative log posterior* $\ell(\mathbf{w}) = \sum_{i=1}^{n} \log(1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i}) + \lambda \mathbf{w}^\top \mathbf{w}$ to find the optimal parameters $\mathbf{w}$.

For a better understanding for the connection of Naive Bayes and Logistic Regression, you may take a peek at these excellent notes.

Summary

Logistic Regression is the discriminative counterpart to Naive Bayes. In Naive Bayes, we first model $P(\mathbf{x}|y)$ for each label y, and then obtain the decision boundary that best discriminates between these two distributions. In Logistic Regression we do not attempt to model the data distribution $P(\mathbf{x}|y)$, instead, we model $P(y|\mathbf{x})$ directly. We assume the same probabilistic form $P(y|\mathbf{x}_i) = \frac{1}{1+e^{-y(\mathbf{w}^T \mathbf{x}_i + b)}}$, but we do not restrict ourselves in any way by making assumptions about $P(\mathbf{x}|y)$ (in fact it can be any member of the Exponential Family). This allows logistic regression to be more flexible, but such flexibility also requires more data to avoid overfitting. Typically, in scenarios with little data and if the modeling assumption is appropriate, Naive Bayes tends to outperform Logistic Regression. However, as data sets become large logistic regression often outperforms Naive Bayes, which suffers from the fact that the assumptions made on $P(\mathbf{x}|y)$ are probably not exactly correct. If the assumptions hold exactly, i.e. the data is truly drawn from the distribution that we assumed in Naive Bayes, then Logistic Regression and Naive Bayes converge to the exact same result in the limit (but NB will be faster).

Multinomial logistic regression

Multnomial logistic regression is the multiclass analog of logistic regression. It does pretty much what you'd expect, except that it allows for more than two classes.