

Bayesian classification and regression

Remember *way* back when we were talking about linear regression. In the probabilistic formulation, we assumed that the response y was given by $w^\top x + \epsilon$, where $\epsilon \sim \mathcal{Nor}(0, \sigma^2)$ was noise. This means that overall, we have:

$$y \mid x, w, \sigma^2 \sim \mathcal{Nor}(w^\top x, \sigma^2) \quad (1)$$

Here, w is an unknown *parameter*. We want to treat it as a *random variable*. Thus, it needs a prior. We use a zero-mean, isotropic Gaussian prior:

$$\begin{aligned} y \mid x, w, \sigma^2 &\sim \mathcal{Nor}(w^\top x, \sigma^2 I) \\ w \mid \lambda &\sim \mathcal{Nor}(0, \lambda^{-1} I) \end{aligned} \quad (2)$$

Here, we have represented the *inverse variance* of the prior as λ (inverse variance is also called *precision*).

Our eventual goal is a prediction distribution, but for now we will work on the posterior of w . Namely, suppose that we observe $(x_1, y_1), \dots, (x_N, y_N)$. We want the distribution:

$$\begin{aligned} p(w \mid x_{1:N}, y_{1:N}, \sigma^2, \lambda) &= \frac{1}{Z} p(w \mid \lambda) \prod_n p(y_n \mid w, x_n, \sigma^2) \\ &= \frac{1}{Z} \mathcal{Nor}(w \mid \lambda) \prod_n \mathcal{Nor}(y_n \mid w^\top x_n, \sigma^2) \\ &= \frac{1}{Z} \mathcal{Nor}(w \mid \lambda) \mathcal{Nor}(Y \mid wX, \sigma^2 I) \end{aligned}$$

We don't immediately know Z , so we cannot explicitly compute this distribution. However, a Gaussian is a conjugate prior to a Gaussian mean, so we believe that this should be available in closed form. It turns out that the posterior is a Gaussian with mean and variance given by:

$$\begin{aligned} \mu^{\text{post}} &= \Sigma^{\text{post}} (\sigma^2 X^\top Y) \\ (\Sigma^{\text{post}})^{-1} &= \lambda I + \sigma^2 X^\top X \end{aligned}$$

If you remember from our discussion of linear regression, the ML weights were given by:

$$w_{ML} = (X^\top X)^{-1} X^\top Y$$

This has the same form (modulo regularization) as the mean of the posterior of the weights. One big advantage of the Bayesian treatment is that we also get the *covariance* of the weights: how *sure* are we about this value. This becomes useful when considering the prediction distribution.

Usually we don't care about the weights themselves, but care only about predictions. I.e., for a test point x_{N+1} , we want to predict y_{N+1} . That is, we want to compute:

$$p(y_{N+1} | x_{1:N+1}, y_{1:N}, \sigma^2, \lambda) = \int dw p(w | x_{1:N}, y_{1:N}, \sigma^2, \lambda) p(y_{N+1} | x_{N+1}, w, \sigma^2)$$

We know that the posterior is Gaussian as is the conditional, so this is:

$$p(y_{N+1} | x_{1:N+1}, y_{1:N}, \sigma^2, \lambda) = \int dw \mathcal{N}(w | \mu^{\text{post}}, \Sigma^{\text{post}}) \mathcal{N}(y_{N+1} | w^\top x_{N+1}, \sigma^2)$$

Again, Gaussian is conjugate to Gaussian so we can compute this in closed form. This yields:

$$p(y_{N+1} | x_{1:N+1}, y_{1:N}, \sigma^2, \lambda) = \mathcal{N}(y_{N+1} | \mu^{\text{post}\top} x, (\sigma^{\text{pred}})^2)$$

$$(\sigma^{\text{pred}})^2 = \frac{1}{\sigma^2} + x_{N+1}^\top \Sigma^{\text{post}} x_{N+1}$$

So, what have we gained over standard linear regression? Not a whole lot – essentially, we get the same predictive *mean*, but now we can compute a variance on the prediction (how accurate do we think we are?). This doesn't seem like a big win. However, an addition win is that we can often do “maximum likelihood II” estimation of the hyperparameters. Remember that usually we have to tune hyperparameters by cross-validation. In Bayesian land, we don't have to. Maximum likelihood II says that we should set the hyperparameters (in this case, σ^2 and λ) to their maximum likelihood values, but where in this case the likelihood contains an integration. I.e., set them as follows:

$$\hat{\sigma}^2, \hat{\lambda} = \underset{\sigma^2, \lambda}{\operatorname{argmax}} \int dw \mathcal{N}(w | 0, \lambda I) \mathcal{N}(X | wX, \sigma^2 I)$$

So although a “true Bayesian” might scoff at ML estimation, it tends to not be too bad for inferring higher-level values. We won't go through this estimation (it's actually not that hard), but since the distributions are conjugate, we get a closed form (Gaussian) for the evidence, and we can take gradients and optimize for the hyperparameters. To me, this is the big win of Bayesian approaches.

Let's consider logistic regression. It will turn out that, due to the presence of the sigmoid function, the conjugacy of the Gaussian won't work out. Here, we have:

$$\begin{aligned} y | x, w &\sim \sigma(w^\top x) \\ w | \lambda^{-1} &\sim \mathcal{N}(0, \lambda^{-1} I) \end{aligned} \tag{3}$$

Here, we have used a zero-mean Gaussian prior with isotropic precision λ as the prior. For N data points, we get:

$$p(w | x_{1:N}, y_{1:N}, \lambda^{-1}) = \frac{1}{Z} \mathcal{N}(w | 0, \lambda^{-1} I) \prod_n \sigma(y_n w^\top x_n) \tag{4}$$

Computing the normalization here is intractable. One approach to solving this would be sampling. Here, we will discuss an alternative approximation technique, the *Laplace* approximation (also called the saddle-point approximation).

There are two interpretations of the Laplace approximation. Some people may find one more amenable than the other, but they both do the same thing:

1. Gaussian interpretation: here, we have some intractable posterior $p(\theta)$; we will approximate it with a Gaussian posterior $q(\theta | \mu, \Sigma)$, where μ and Σ are chosen so that the approximation is as “close” as possible.
2. Taylor series interpretation: we still have intractable $p(\theta)$; we will construct a Taylor series approximation to $\log p$, centered at its mode. We’ll truncate the Taylor series approximation at the quadratic term.

The fact that we truncate the Taylor series at the quadratic term means that we only have a zero-th, first and second order term. Since we center the Taylor series at the mode, this means that the first order term (the gradient) is zero. This yields an expression that includes only zeroth and second order terms – exactly a Gaussian.

Let’s start with a simple case: θ is univariate. We have $p(\theta) = \frac{1}{Z}f(\theta)$, where f is unnormalized and $Z = \int d\theta f(\theta)$. We’d ideally like to compute Z .

We’re going to approximate f by a Gaussian. Why? Because we *can* compute the normalizing constant of a Gaussian. We begin by computing the mean of the approximating Gaussian, which we’ll place at the mode of f . Thus, we have to compute:

$$\theta_0 = \underset{\theta}{\operatorname{argmax}} f(\theta)$$

This is, in probabilistic models, just the MAP value of θ .

Next, we make a second order approximation:

$$\log f(\theta) \approx \log f(\theta_0) - \frac{1}{2}A(\theta - \theta_0)^2$$

$$a = \left. \frac{\partial^2}{\partial \theta^2} \log f(\theta) \right|_{\theta=\theta_0}$$

There are two steps for doing the Laplace approximation. First, we compute θ_0 as the MAP value of θ . Then we compute a as a second derivative of θ , evaluated at θ_0 .

Given this, we approximate f itself as:

$$f(\theta) \approx f(\theta_0) \exp \left[-\frac{a}{2}(\theta - \theta_0)^2 \right]$$

But this f is just a Gaussian, so we compute:

$$Z = p(\theta_0) \left(\frac{2\pi}{a} \right)^{1/2}$$

Now we move to the multivariate case. This is essentially the same, but with vectors and matrices. We start with the approximation:

$$\begin{aligned}\log f(\theta) &\approx \log f(\theta_0) - \frac{1}{2}(\theta - \theta_0)^\top Z(\theta - \theta_0)^\top \\ \theta_0 &= \operatorname{argmax}_\theta f(\theta) \\ A &= -\nabla\nabla \log f(\theta)|_{\theta=\theta_0} \\ A_{ij} &= \frac{\partial^2}{\partial\theta_i\partial\theta_j} \log f(\theta)\Big|_{\theta=\theta_0}\end{aligned}$$

This again leads to a Gaussian approximation, with:

$$q(\theta) = \left[\frac{\det A}{(2\pi)^M} \right]^{1/2} \exp \left[-\frac{1}{2}(\theta - \theta_0)^\top A(\theta - \theta_0) \right] = \mathcal{Nor}(\theta \mid \theta_0, A^{-1})$$

So the story is the same in the multivariate case. Compute the vector θ_0 that maximizes f (again, the MAP solution). Then compute A as a matrix of second derivatives. This yields a Gaussian approximation.

To apply the Laplace approximation to logistic regression, we have to follow two steps. First, we compute the MAP solution for the weights to obtain w_0 . We already know how to compute the MAP (CS6350 students did it in HW1). The next step is to compute the matrix of second derivatives (aka, the *Hessian*). This is given by:

$$A = \lambda + \sum_n y_n(1 - y_n)x_n x_n^\top$$

Thus, we obtain $q(w) = \mathcal{Nor}(w \mid w_0, A)$.

Of course, we rarely care about w itself, but rather want to make predictions. This is, again, intractable. The standard solution (described in detail in the book) yields:

$$\begin{aligned}p(y_{N+1} = 1 \mid x_{1:N+1}, y_{1:N}, \lambda) &= \int dw \sigma(w^\top x_{N+1}) p(w \mid x_{1:N}, y_{1:N}, \lambda) \\ &\approx \int dw \sigma(w^\top x_{N+1}) \mathcal{Nor}(w \mid w_0, A) \\ &\approx \sigma \left(\left[1 + \frac{\pi}{8} x_{N+1}^\top A x_{N+1} \right]^{-1/2} w_0 x_{N+1} \right)\end{aligned}$$

Note that the first term here is always positive, and the second term is just the standard MAP prediction. The first term serves to scale the uncertainty – if A is very uncertain, this value will be small and hence the resulting probability will be close to 0.5.