Bayesian inference and prediction

The previous note introduced a probabilistic model for regression. Nothing changed when we did maximum likelihood: we still did least squares. However, Bayesian inference on this model reveals how uncertain we are about the underlying model. We no longer fit a single model, instead we constructed a distribution over all possible models. Now we'll look at how to use that distribution. But first, a simple concrete prediction problem as a work-up exercise.

A toy prediction problem

In lectures I'll show you 3 cards, each with two sides: Card 1 has one white and one black side, card 2 has two black sides, and card 3 has two white sides.

I shuffle the cards and turn them over randomly. I select a card and way-up uniformly at random and place it on a table.

Question[1]**:** You see a black side. What is the probability that the other side of the same card is white?

$$P(x_2 = \mathtt{W} \,|\, x_1 = \mathtt{B}) = \quad 1/3, \quad 1/2, \quad 2/3, \quad \text{other?}$$

Hopefully the process by which a card is selected should be clear: $P(c) = 1/3$ for $c = 1, 2, 3$, and the side you see first is chosen at random: e.g., $P(x_1 = \mathtt{B} \,|\, c = 1) = 0.5$.

Many people get this puzzle wrong on first viewing (it's easy to mess up). Although regardless of whether the answer is obvious or not, it's a simple example on which to demonstrate some formalism.

When I first tried to use probabilities for prediction problems, I often found myself going in circles using Bayes' rule:

$$P(x_2 = \mathtt{W} \,|\, x_1 = \mathtt{B}) \;=\; \frac{\boxed{P(x_1 = \mathtt{B} \,|\, x_2 = \mathtt{W})}\, P(x_2 = \mathtt{W})}{P(x_1 = \mathtt{B})}$$

The boxed term is no more obvious than the answer! Bayes' rule can't be used blindly to obtain any conditional probability. It is used specifically to 'invert' processes that we understand. The first step to solve inference problems is to write down a model of the data.

A model of the card game states that we first picked one of the three cards uniformly at random:

$$P(c) = \begin{cases} 1/3 & c = 1, 2, 3 \\ 0 & \text{otherwise.} \end{cases}$$

Then given the card, we picked one of the faces at random:

$$P(x_1 = \mathtt{B} \,|\, c) = \begin{cases} 1/2 & c = 1 \\ 1 & c = 2 \\ 0 & c = 3 \end{cases}$$

Bayes rule can 'invert' this process to tell us $P(c \,|\, x_1 = \mathtt{B})$. It infers a posterior distribution over explanations of how we got the data that we observed.

---

1. This card problem is Ex. 8.10a), MacKay, p142. It is not the same as the famous "Monty Hall" or "three doors" puzzle (MacKay Ex. 3.8–9). The Monty Hall problem is also worth understanding. Although the card problem is (hopefully) less controversial and more straightforward. Also fewer students have seen it before and memorized the answer.

Inferring the card:

$$P(c \mid x_1 = \mathtt{B}) = \frac{P(x_1 = \mathtt{B} \mid c)\, P(c)}{P(x_1 = \mathtt{B})}$$
$$\propto \begin{cases} 1/2 \cdot 1/3 = 1/6 & c = 1 \\ 1 \cdot 1/3 = 1/3 & c = 2 \\ 0 & c = 3 \end{cases}$$
$$= \begin{cases} 1/3 & c = 1 \\ 2/3 & c = 2. \end{cases}$$

This unbalanced posterior distribution over cards doesn't match some people's intuition. "Aren't there two possible cards given a black side, so it's 50–50?". While there are two possible cards, the *likelihood* for one of these options is $2\times$ larger. Analogy: I pick a dice at random from a 6-sided dice, a 10-sided dice and a 100-sided dice. I roll the dice and tell you I got an 8. You know it's not the 6-sided dice, so there are two options, the 10-sided dice or the 100-sided dice. But the 10-sided dice is more likely.

We can now spot the answer to the card problem. The other side will be white only if it's card 1, and we have now computed the probability representing our belief that we're looking at card 1. For more complex problems we want a more mechanical approach.

The probability for predicting the next outcome that we want is $P(x_2 \mid x_1 = \mathtt{B})$. We need to introduce the card c into the maths, so that we can use our model and its posterior distribution. We use the sum rule to introduce the card choice c as a dummy variable, and then split up the joint probability with the product rule:

$$P(x_2 \mid x_1 = \mathtt{B}) = \sum_{c \in 1,2,3} P(x_2, c \mid x_1 = \mathtt{B})$$
$$= \sum_{c \in 1,2,3} P(x_2 \mid x_1 = \mathtt{B}, c)\, P(c \mid x_1 = \mathtt{B}).$$

This equation says that we consider each of the predictions that we would make if we knew the card, and weight each prediction by the posterior probability of that card. (And adding up the options we get $1/3$.)

General strategy for solving prediction problems:

When we want to predict some quantity y, we often find that we can't immediately write down mathematical expressions for $P(y \mid \text{data})$.

So we introduce "stuff" z, model parameters and/or latent variables, that helps us define the problem, by using the sum rule:

$$P(y \mid \text{data}) = \sum_z P(y, z \mid \text{data}).$$

If the stuff is real-valued, we'll have an integral over a PDF instead of a sum over probabilities. Either way, we split up the joint probability using the product rule:

$$P(y \mid \text{data}) = \sum_z P(y \mid z, \text{data})\, P(z \mid \text{data}).$$

If we would know how to predict y if we were told the extra stuff z, we can find the predictive distribution: weight the predictions for each possible stuff, $P(y \mid z, \text{data})$, by the posterior probability of the stuff, $P(z \mid \text{data})$. We get the posterior from Bayes' rule.

Sometimes the extra stuff summarizes everything we need to know to make a prediction: $P(y \mid z, \text{data}) = P(y \mid z)$. Although not in the card game example above.

Not convinced?

Not everyone believes the answer to the card game question. Sometimes probabilities are counter-intuitive. Here is an Octave/Matlab simulator I wrote for the card game question:

```
cards = [1 1;
         0 0;
         1 0];
num_cards = size(cards, 1);

N = 0; % Number of times first side is black
kk = 0; % Out of those, how many times the other side is white

for trial = 1:1e6
    card = ceil(num_cards * rand());
    side = 1 + (rand < 0.5);
    other_side = (side==1) + 1;
    x1 = cards(card, side);
    x2 = cards(card, other_side);

    if x1 == 0
        N = N + 1; % just seen another black face
        kk = kk + (x2 == 1); % count if other side was white
    end
end

approx_probability = kk / N
```

An implementation without the for loop would probably be faster, but here I wanted to convince you, so didn't want the code to be at all cryptic.

Predictions for Bayesian linear regression

Compared to the statistics literature, the machine learning literature places a greater emphasis on prediction than inferring unknown parameters. Here we explore how to make a Bayesian prediction given a posterior over the parameters.

The prediction of an unknown output y at a test location $\mathbf{x}$ is expressed as a probability distribution. We condition on training data $\mathcal{D} = \{\mathbf{x}^{(n)}, y^{(n)}\}$. We want to refer to our model, in particular its regression weights, so we introduce them using the sum rule:

$$p(y \,|\, \mathbf{x}, \mathcal{D}) = \int p(y, \mathbf{w} \,|\, \mathbf{x}, \mathcal{D}) \, \mathrm{d}\mathbf{w}.$$

Then we split up the joint probability using the product rule:

$$p(y \,|\, \mathbf{x}, \mathcal{D}) = \int p(y \,|\, \mathbf{x}, \mathbf{w}) \, p(\mathbf{w} \,|\, \mathcal{D}) \, \mathrm{d}\mathbf{w}.$$

Here the first term isn't conditioned on the data, because if we know the parameters, the data doesn't help make the prediction. The second term isn't conditioned on the test input location, because we assume the test location doesn't tell us anything about the weights[2].

What are the probabilities in this integral? As explained in the previous note, the posterior over the weights is Gaussian for a linear regression model with Gaussian noise. Following Murphy's notation, I'll write:

$$p(\mathbf{w} \,|\, \mathcal{D}) = \mathcal{N}(\mathbf{w}; \mathbf{w}_N, V_N),$$

2. Non-examinable: There are *transductive* learning systems that do base inferences on the locations of test inputs.

where $\mathbf{w}_N$ is the posterior mean after observing N datapoints, and V_N is the posterior covariance. The other term in the integrand is the predictive distribution for known parameters. That is, a Gaussian with noise variance σ_y^2 centred around the function value for the test input:[3]

$$p(y \mid \mathbf{x}, \mathbf{w}) = \mathcal{N}(y;\, \mathbf{w}^\top \mathbf{x}, \sigma_y^2).$$

So now we 'just' need to do the integral and we have our answer.

Only a little probability theory is needed for Bayesian inference and prediction in this course: just the sum and product rule. We follow the rules in the general strategy above. However, actually solving the sums or integrals required is usually hard.

Linear and Gaussian models are one of the rare cases where we can do the integrals in closed form. Multiplying the terms in the integrand together, and carefully tracking terms we can factor out a Gaussian in $\mathbf{w}$ that integrates to 1. A Gaussian in y remains. It's some work, but can be done.

Anticipating that our predictive distribution is Gaussian, we can look for an easier way to identify its mean and variance. The test output is a noisy version of the underlying function:

$$y = f(\mathbf{x}) + \nu = \mathbf{x}^\top \mathbf{w} + \nu, \quad \nu \sim \mathcal{N}(0, \sigma_y^2).$$

The function value is a linear transformation of the weights, so our beliefs about the function value are Gaussian distributed with mean:

$$\mu_f = \mathbb{E}[\mathbf{x}^\top \mathbf{w}] = \mathbf{x}^\top \mathbf{w}_N$$

and variance[4]

$$\mathrm{var}[f] = \mathbf{x}^\top V_N \mathbf{x}.$$

So our belief about the function value is:

$$p(f \mid \mathcal{D}, \mathbf{x}) = \mathcal{N}(f;\, \mathbf{x}^\top \mathbf{w}_N,\, \mathbf{x}^\top V_N \mathbf{x}).$$

Adding Gaussian noise, our belief about a new output is also Gaussian. The independent zero mean noise doesn't change the mean, but adds σ_y^2 to the variance:

$$p(y \mid \mathcal{D}, \mathbf{x}) = \mathcal{N}(y;\, \mathbf{x}^\top \mathbf{w}_N,\, \mathbf{x}^\top V_N \mathbf{x} + \sigma_y^2).$$

This answer matches the answer quoted for the integral by Murphy.

Decision making

Given a test input $\mathbf{x}$, the Bayesian approach says that we don't know what the output y will be, and so doesn't return a single value. However, in most prediction competitions (such as a Kaggle competition, but not always[5]) we need to provide a point-estimate or guess of the output, $\hat{y}$. In real-world applications, we need to make decisions. Usually each guess or decision should be chosen to minimize our expected loss, then over many guesses our average performance will be good.

For a given application we need to write down a loss function $L(y, \hat{y})$. The loss says how bad it is to guess $\hat{y}$ if the actual output is y. Our expected loss given training data $\mathcal{D}$ give us our cost function:

$$c = \mathbb{E}_{p(y \mid \mathbf{x}, \mathcal{D})}[L(y, \hat{y})] = \int L(y, \hat{y})\, p(y \mid \mathbf{x}, \mathcal{D})\, \mathrm{d}y.$$

---

3. I'm assuming that the noise level σ_y^2 is part of our general background knowledge. I won't consider different values or infer σ_y^2 in this note, so I haven't bothered to explicitly condition on it in any of the probabilities.
4. $\mathrm{var}[f] = \mathbb{E}[(f - \mu_f)(f - \mu_f)] = \mathbb{E}[\mathbf{x}^\top (\mathbf{w} - \mathbf{w}_N)(\mathbf{w} - \mathbf{w}_N)^\top \mathbf{x}] = \mathbf{x}^\top \mathrm{cov}[\mathbf{w}]\mathbf{x}$
5. https://link.springer.com/chapter/10.1007/11736790_1

For square loss, $L(y, \hat{y}) = (y - \hat{y})^2$, we can differentiate the cost with respect to our point-estimate:

$$\frac{\partial c}{\partial \hat{y}} = \mathbb{E}_{p(y \mid \mathbf{x}, \mathcal{D})}[-2(y - \hat{y})] = -2(\mathbb{E}_{p(y \mid \mathbf{x}, \mathcal{D})}[y] - \hat{y}).$$

Setting the derivative to zero, the optimal guess is the posterior mean, $\hat{y} = \mathbb{E}_{p(y \mid \mathbf{x}, \mathcal{D})}[y]$. For Bayesian linear regression with a Gaussian prior and noise model, it can be shown that the posterior mean corresponds to using an L2 regularized model. If all we care about is mean squared error, the Bayesian approach doesn't change what we do.

However, the uncertainty can still be useful. If we identify predictions that are uncertain, it could warn us to manually intervene, or gather data relevant to those cases. That is, making a single point estimate is often not the only decisions we make based on a model. Moreover, in real-world applications, sensible loss functions are often asymmetric.

Imagine that a bakery makes an estimate $\hat{y}$ of tomorrow's bread sales y. If it were possible to make perfect predictions, so we knew that $\hat{y} = y$, then we would bake exactly $\hat{y}$ loaves of bread. We'd sell all our bread, and send no customers away. Sadly we can't make perfect predictions. My local baker sells out nearly every day. Presumably they really don't like throwing bread away, so set $\hat{y}$ smaller than the average possible value of y. Other bakers seem to attach a different loss to waste, and make decisions that regularly result in throwing bread away[6].

As another example, a business-to-business supplier of non-perishable goods will pay warehouse fees to keep excess stock, if failing to fulfill orders will lose customers in the long term. For them, the cost of underestimating sales is much higher than the cost of overestimating.

The Bayesian approach separates modelling data from the application-specific loss function. Multiple decisions, with different losses, can be made from the same model. An alternative and popular approach, "empirical risk minimization", fits a model function to the training data to directly optimize an application-specific loss function.[7]

Further Reading

Murphy covers prediction for Bayesian linear regression: Section 7.6.2 and Figure 7.12.

Murphy Section 5.7–, and MacKay Chapter 36 (p451), have more detail on Bayesian decision making.

Check your understanding

MacKay's book has more toy prediction exercises like the card game: Ex. 3.12 and 3.14, p58. You might have intuitive answers, but try to solve them mechanically by writing down an explicit model and applying rules of probability.

What happens to the uncertainty of predictions as $\mathbf{x}$ grows large. Why?

More detail for those who love linear algebra

A brute force way to get the predictive distribution for linear regression is to write the integrand as a joint distribution, as a standard multivariate Gaussian using block matrices:

$$p(\mathbf{w}, y \mid \mathbf{x}, \mathcal{D}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{w} \\ y \end{bmatrix}; \begin{bmatrix} \mathbf{w}_N \\ m \end{bmatrix}, \begin{bmatrix} V_N & \Sigma_{\mathbf{w},y} \\ \Sigma_{y,\mathbf{w}} & r^2 \end{bmatrix}\right).$$

6. Or if more responsible, they find someone to take the bread who wouldn't purchase it anyway.
7. Warning: You will find dogmatic advocacy for each of these approaches over the other. Personally, I've used both approaches. As usual, the optimal decision depends upon the circumstances.

Written in this form, the marginal mean and variance of y are immediately available, and we'd write $p(y \mid \mathbf{x}, \mathcal{D}) = \mathcal{N}(y; m, r^2)$. But if you find that intimidating, I agree! Examining the quadratic form involving y and $\mathbf{w}$ in the joint distribution gives us the joint inverse covariance, and we need to do quite a few lines of linear algebra to identify $m = \mathbf{x}^\top \mathbf{w}_N$ and $r^2 = \mathbf{x}^\top V_N \mathbf{x} + \sigma_y^2$. That seems like a lot of work. Sources like the matrix cookbook can help. But in this case there was an easier way.