

# Bayes for Regression

Winter 2010

This is a document about the bias-variance tradeoff and a maximum likelihood explanation for least-squares regression.

## 1 Least Squares as maximum likelihood

Assume we have a set of  $n$  training points  $\mathcal{X} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_n, y_n)\}$  for a regression problem, so each  $y_i \in \mathfrak{R}$ . Assume the data is labeled by some function  $f^*$  in a function class  $\mathcal{F}$  (such as the linear functions) with some zero mean gaussian noise with standard deviation  $\sigma$ . Thus each  $y_i = f^*(\mathbf{x}_i) + \epsilon_i$  where the  $\epsilon_i$  are independent and  $p(\epsilon_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-\epsilon_i^2}{2\sigma^2}\right)$ . Note that we are assuming noise only on the  $y_i$  values and not on the  $\mathbf{x}_i$  values.

Now consider any  $f \in \mathcal{F}$ , and the probability (density) of  $f$  generating the  $y_i$  values from observed  $\mathbf{x}_i$ 's in the data. Denote this  $p(\mathcal{X}|f)$  even though it also depends on the  $\mathbf{x}_i$  values (this is like assuming the  $x_i$  values are fixed in advance). With the assumption that the noise is iid gaussian,

$$\begin{aligned} p(\mathcal{X}|f) &= \prod_{i=1}^n p(y_i|f, \mathbf{x}_i) \\ &= \prod_{i=1}^n p(\epsilon_i = y_i - f(x_i)) \\ &= \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(y_i - f(x_i))^2}{2\sigma^2}\right) \end{aligned}$$

We can now take natural logs to turn the product into a sum.

$$\begin{aligned} \log p(\mathcal{X}|f) &= \sum_{i=1}^n \left( -\log(\sqrt{2\pi}\sigma) - \frac{(y_i - f(x_i))^2}{2\sigma^2} \right) \\ &= -n \log(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - f(x_i))^2 \end{aligned}$$

The first term is a constant for all  $f$ , and the  $f \in \mathcal{F}$  minimizing the sum will be the  $f$  that maximizes the likelihood of the data. Furthermore, the  $f$  minimizing the sum is precisely the  $f$  minimizing the squared error on the data.

Note that due to the random noise, the maximum likelihood  $f$  need not be the function  $f^*$  generating the data.

## 2 Bias Variance Decomposition

Assume we have a set of  $n$  randomly drawn training points  $\mathcal{X} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_n, y_n)\}$  for a regression problem, so each  $y_i \in \mathfrak{R}$ . As before, assume the data is labeled by some function  $f^*$  with some zero mean gaussian noise with standard deviation  $\sigma$ . Thus each  $y_i = f^*(\mathbf{x}_i) + \epsilon_i$  where the  $\epsilon_i$  are independent and  $p(\epsilon_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-\epsilon_i^2}{2\sigma^2}\right)$ .

Now assume that we use a learning method to find a function  $f_{\mathcal{X}}$  from the  $n$  training points. Most algorithms choose different  $f_{\mathcal{X}}$  when the  $y_i$  values differ. So the function  $f_{\mathcal{X}}$  depends on the "experiment" we are performing (the random drawing of the sample).

Now consider using the learned  $f_{\mathcal{X}}$  to predict on a new point  $(\mathbf{x}_0, y_0)$  where  $y_0 = f^*(\mathbf{x}_0) + \epsilon_0$ . We can write the squared error of  $f_{\mathcal{X}}(x_0)$  as follows (using  $\overline{f_{\mathcal{X}}(\mathbf{x}_0)}$  for the value of  $f_{\mathcal{X}}(x_0)$  averaged over the randomly drawn sample and  $\epsilon_0$ ):

$$(f_{\mathcal{X}}(\mathbf{x}_0) - y_0)^2 = (f_{\mathcal{X}}(\mathbf{x}_0) - (f^*(\mathbf{x}_0) + \epsilon_0))^2 \quad (1)$$

$$= (f_{\mathcal{X}}(\mathbf{x}_0) - \overline{f_{\mathcal{X}}(\mathbf{x}_0)} + \overline{f_{\mathcal{X}}(\mathbf{x}_0)} - f^*(\mathbf{x}_0) - \epsilon_0)^2 \quad (2)$$

$$= (f_{\mathcal{X}}(\mathbf{x}_0) - \overline{f_{\mathcal{X}}(\mathbf{x}_0)})^2 + (\overline{f_{\mathcal{X}}(\mathbf{x}_0)} - f^*(\mathbf{x}_0))^2 + \epsilon_0^2 \quad (3)$$

$$+ 2(f_{\mathcal{X}}(\mathbf{x}_0) - \overline{f_{\mathcal{X}}(\mathbf{x}_0)})(\overline{f_{\mathcal{X}}(\mathbf{x}_0)} - f^*(\mathbf{x}_0) - \epsilon_0) - 2\epsilon_0(\overline{f_{\mathcal{X}}(\mathbf{x}_0)} - f^*(\mathbf{x}_0)) \quad (4)$$

We next take the expectation with respect to both the randomly drawn  $\mathcal{X}$  and  $\epsilon_0$ . When we do this, the first term on line 4 drops out as  $E[f_{\mathcal{X}}(\mathbf{x}_0) - \overline{f_{\mathcal{X}}(\mathbf{x}_0)}] = \overline{f_{\mathcal{X}}(\mathbf{x}_0)} - \overline{f_{\mathcal{X}}(\mathbf{x}_0)} = 0$ . Similarly, the second term on line 4 also goes away as  $E[\epsilon_0] = 0$  since  $\epsilon_0$  is a draw from a zero-mean Gaussian. This leaves the following.

$$\begin{aligned} E[(f_{\mathcal{X}}(\mathbf{x}_0) - y_0)^2] &= E[(f_{\mathcal{X}}(\mathbf{x}_0) - \overline{f_{\mathcal{X}}(\mathbf{x}_0)})^2 + (\overline{f_{\mathcal{X}}(\mathbf{x}_0)} - f^*(\mathbf{x}_0))^2 + \epsilon_0^2] \\ &= E[(f_{\mathcal{X}}(\mathbf{x}_0) - \overline{f_{\mathcal{X}}(\mathbf{x}_0)})^2] + E[(\overline{f_{\mathcal{X}}(\mathbf{x}_0)} - f^*(\mathbf{x}_0))^2] + E[\epsilon_0^2] \\ &= E[(f_{\mathcal{X}}(\mathbf{x}_0) - \overline{f_{\mathcal{X}}(\mathbf{x}_0)})^2] + (\overline{f_{\mathcal{X}}(\mathbf{x}_0)} - f^*(\mathbf{x}_0))^2 + \sigma^2 \end{aligned}$$

Each of the the three terms in the previous equation have an interesting interpretation.

- $E[(f_{\mathcal{X}}(\mathbf{x}_0) - \overline{f_{\mathcal{X}}(\mathbf{x}_0)})^2]$  is the variance of  $f_{\mathcal{X}}(\mathbf{x}_0)$  with respect to the randomly drawn training data.
- $\overline{f_{\mathcal{X}}(\mathbf{x}_0)} - f^*(\mathbf{x}_0)$  is how much the average prediction (over random training sets) differs from the average value at  $x_0$ . This is the *bias* of the learning method (on test instance  $\mathbf{x}_0$ ). The second term,  $(\overline{f_{\mathcal{X}}(\mathbf{x}_0)} - f^*(\mathbf{x}_0))^2$  is the square of the bias.
- $\sigma^2$  is the random variance of  $y_0$ . Even if we knew  $f^*(\mathbf{x}_0)$ , we could not eliminate this term, so it is sometimes called the *intrinsic error*.

We can now re-write the expected squared error as follows.

$$E[(f_{\mathcal{X}}(\mathbf{x}_0) - y_0)^2] = \text{variance} + \text{bias}^2 + \text{intrinsicError}$$

We performed the bias-variance decomposition with respect to a particular test instance  $\mathbf{x}_0$ . However, it is possible to integrate over  $\mathbf{x}_0$  and get a similar result when the test instance (and training instances) are randomly selected from some distribution over the domain of feature vectors.

The bias-variance decomposition shows that minimizing a learning algorithm's bias and/or variance will improve the (expected) generalization error of its hypotheses. However, reducing bias means allowing the function to be more flexible and more closely fit the data, which tends to increase the variance. Similarly, reducing the variance usually increases the bias. This is called the bias-variance tradeoff, and one way of comparing algorithms is to locate their relative positions on the bias-variance tradeoff. For example, nearest neighbor produces relatively unstructured hypotheses with low bias and high variance while the highly structured hypotheses produced by the Perceptron algorithm tend to have high bias and lower variance.