Statistical Methods: Portfolio 2

By Henry Bourne

1 Bias-Variance Decomposition

What is the mathematical explanation of overfitting? Why is cross-validation a good measurement of the generalization of a prediction? we introduce a frequentist analysis to answer these questions.

First we will assume an outcome, y_i, is generated by $y_i = g(\boldsymbol{x}_i) + \epsilon_i$. Where $g_{\boldsymbol{x}} : \mathbb{R}^d \to \mathbb{R}$ is a deterministic function, $\forall i, \epsilon_i$ independent of $\boldsymbol{x}_i$ and $\mathbb{E}(\epsilon_i) = 0$. For now for the sake of simplicity we also assume $\boldsymbol{x}_i$ are fixed. We also partition the dataset, D, into training, D_0, and testing, D_1 datasets and will use least squares as our error/cost function, denoted E.

Whats better than the testing error on a specific dataset? One answer is the expectation of the testing error over the whole dataset. ie. $\mathbb{E}_D(E(D_1, \boldsymbol{w}_{LS})) = \mathbb{E}_D(\sum_i [y_i - f(\boldsymbol{x}_i; \boldsymbol{w}_{LS})]^2) = \sum_i \mathbb{E}_D([y_i - f(\boldsymbol{x}_i; \boldsymbol{w}_{LS})]^2 | \boldsymbol{x}_i)$. We can decompose this as follows (proof in section A.1):

Definition 1.1 ***Bias-Variance Decomposition:***

$$\mathbb{E}_D([y_i - f(\boldsymbol{x}_i; \boldsymbol{w}_{LS})]^2 | \boldsymbol{x}_i) = Var(\epsilon_i) + [g(\boldsymbol{x}_i) - \mathbb{E}(f(\boldsymbol{x}_i; \boldsymbol{w}_{LS}))|\boldsymbol{x}_i]^2 + Var(f(\boldsymbol{x}_i; \boldsymbol{w}_{LS})|\boldsymbol{x}_i) \quad (1)$$

The 1^{st} term measures the randomness of our data generating process (this is beyond our control), the 2^{nd} term shows the accuracy of our expected prediction and the 3^{rd} term shows how easily our fitted prediction function is affected by the randomness of the dataset. When we are using a polynomial feature transform and increase b, the prediction function becomes more complex and hence the bias decreases, but conversely the variance increases. Note that we use the $\boldsymbol{w}_{LS}$ estimator here but the bias-variance decomposition holds true for any estimator, furthermore when we do use $\boldsymbol{w}_{LS}$ the bias equals zero. Also note that we can derive bias-variance Decompositions for different loss functions. A balance between trying to minimize the bias and variance gives us the **minimum expected error**.

1.1 Justifying K-fold cross-validation

First we define the:

Definition 1.2 ***In-Sample Error:***
This is the collective error of the training set, we can calculate it by averaging the expected error obtained over all the $\boldsymbol{x}_i$'s in the training set,

$$\frac{1}{n}\sum_{i=1}^{n} \mathbb{E}((y_i - f(\boldsymbol{x}_i; \boldsymbol{w}_{LS}))^2 | \boldsymbol{x}_i) \quad (2)$$

In practice the in-sample error is not useful as we cannot calculate it, because we do not know $g(\boldsymbol{x})$ or the distribution of the ϵ_i, instead we use the:

Definition 1.3 ***Out-sample Error:***
The error over the whole distribution of $\boldsymbol{x}$,

$$\mathbb{E}_{\boldsymbol{x}}\mathbb{E}[(\boldsymbol{y} - f(\boldsymbol{x}; \boldsymbol{w}_{LS}))^2 | \boldsymbol{x}] \quad (3)$$

Now that $\boldsymbol{x}$ is being treated as a random quantity, we have,

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{x}}\mathbb{E}[(\boldsymbol{y} - f(\boldsymbol{x}; \boldsymbol{w}_{LS}))^2 | \boldsymbol{x}] &= \mathbb{E}_{\boldsymbol{x}}\mathbb{E}_{D_1}\mathbb{E}_{D_0}[(y - f(\boldsymbol{x}; \boldsymbol{w}_{LS}))^2 | \boldsymbol{x}] & (4)\\
&= \mathbb{E}_{p(\boldsymbol{x})}\mathbb{E}_{p(y|\boldsymbol{x})}\mathbb{E}_{D_0}[(y - f(\boldsymbol{x}; \boldsymbol{w}_{LS}))^2] & (5)\\
&= \mathbb{E}_{D_0}\mathbb{E}_{p(y,\boldsymbol{x})}[(y - f(\boldsymbol{x}; \boldsymbol{w}_{LS}))^2] & (6)
\end{aligned}$$

Now, can we approximate the out-sample error? Suppose we have datasets $D^{(1)}, ..., D^{(K)}$ containing pairs $(y, \boldsymbol{x})$ from $p(y, \boldsymbol{x})$. Let $D^{(k)} := D_0^{(k)} \cup D_1^{(k)}$, the following holds under mild conditions (conditions that aren't too hard to obtain):

$$\mathbb{E}_{D_0}\mathbb{E}_{p(y,\boldsymbol{x})}[(y - f(\boldsymbol{x}; \boldsymbol{w}_{LS}))^2] \approx \frac{1}{K} \sum_{k=1,...,K} \frac{1}{n_k} \sum_{(y,\boldsymbol{x}) \in D_1^{(k)}} (y - f(\boldsymbol{x}; \boldsymbol{w}_{LS})^{(k)})^2 \tag{7}$$

Where $f(\boldsymbol{x}; \boldsymbol{w}_{LS})^{(k)}$ denotes the prediction function trained on $D_0^{(k)}$ and where n_k is the size of $D_1^{(k)}$. If we suppose that $D_0^{(k)}$ is the k-th split of an iid. dataset and $D_1^{(k)}$ is the rest of the dataset, then the above result provides some justification for the use of K-fold cross-validation.

2 Feature Transforms and Kernel Methods

We will start by discussing different types of feature transforms before diving into kernel methods.

2.1 Feature transforms

We've already seen polynomial feature transforms, but why do they work? Let's get some intuition from the the 1-dimensional case. The taylor series of the data generating function, g, at zero is: $g(x) = g(0)(x-0)^0 + g'(0)(x-0)^1 + \frac{g''(0)}{2!}(x-0)^2 + ...$ The taylor series tells us that we can approximate a smooth function using polynomial terms (at some cost), this gives some intuition for how a polynomial transform could allow us to create a good model of the data generating function. Another way we can decompose g is using the **Fourier series**: $g(x) = a_0 + \sum_{i=1}^{\infty}[a_i \sin(ix) + b_i \cos(ix)]$. We can use the fourier series as intuition for another type of transform:

Definition 2.1 ***Trigonometric transform:***
Used to approximate g(x) over the time domain, defined as follows,

$$\phi(\boldsymbol{x}) := [\sin(\boldsymbol{x}), \cos(\boldsymbol{x}), sin(2\boldsymbol{x}), cos(2\boldsymbol{x}), ..., \sin(b\boldsymbol{x}), \cos(b\boldsymbol{x})] \tag{8}$$

polynomial and trigonometric transforms are based on the idea a function can be approximated by a:

Definition 2.2 ***Linear basis expansion:***
A linear basis expansion of $g(\boldsymbol{x})$ is,

$$g(\boldsymbol{x}) \approx f(\boldsymbol{x}; \boldsymbol{w}) = \langle \boldsymbol{w}, \phi(\boldsymbol{x}) \rangle = \sum_{i=1}^{b} w^{(i)} \phi^{(i)}(\boldsymbol{x}) \tag{9}$$

where $\phi^{(i)}$ are called ***basis functions***

A widely used basis function for regression tasks is the,

Definition 2.3 ***Radial Basis Function (RBF):***

$$\phi^{(i)}(\boldsymbol{x}) := \exp(-\frac{||\boldsymbol{x} - \boldsymbol{x}_i||^2}{2\sigma^2}) \tag{10}$$

where $\sigma > 0$ is called the ***bandwidth****. Note that σ is determined before fitting, a common practice is setting it equal to the median of all pairwise distances of $\boldsymbol{x}$ in your dataset.*
$\boldsymbol{x}_i$ are called ***RBF centroids*** *(often chosen randomly from dataset). We will use $\boldsymbol{\phi}(\boldsymbol{x})$ to denote $[\phi^{(1)}(\boldsymbol{x}), ..., \phi^{(b)}(\boldsymbol{x})]$*

If we imagine the prediction function starting as a flat line, if $w^{(i)} > 0$ then the RBF at at $\boldsymbol{x}_i$ lifts up the value of the prediction function around $\boldsymbol{x}_i$, we have the converse is true if $w^{(i)} < 0$. Each RBF defines a ball on which the prediction function is supported (with radius σ), if $g(\boldsymbol{x})$ has a wide support then the prediction function must be supported almost everywhere and need many centroids. The packing number (of the number of balls) $b = O(c^d)$, where d is dimension of $\boldsymbol{x}$.

2.2 Kernel methods

The larger the value of b, the higher the dimensionality of the feature space and the more flexible our prediction function is. Can we have an ∞-dimensional feature space?

Suppose $\phi(\boldsymbol{x})$ maps $\boldsymbol{x}$ to an ∞-dim. feature space, we will have that $\boldsymbol{w}$ will be ∞ly long. However, we can show that $\boldsymbol{w}_{LS-R} := (\boldsymbol{\phi}\boldsymbol{\phi}^T + \lambda\boldsymbol{I})^{-1}\boldsymbol{\phi}\boldsymbol{y}^T = \boldsymbol{\phi}(\boldsymbol{\phi}^T\boldsymbol{\phi} + \lambda\boldsymbol{I})^{-1}\boldsymbol{y}^T$, where $\boldsymbol{\phi}$ is short for $\phi(\boldsymbol{X})$ (proof in section A.2). We now no longer need to compute $\boldsymbol{\phi}\boldsymbol{\phi}^T$ (intractable), we instead can compute $\boldsymbol{\phi}^T\boldsymbol{\phi} \in \mathbb{R}^{n*n}$.

Is there a way we can compute the inner-product without computing $\phi(\cdot)$? Define $k(\boldsymbol{a}, \boldsymbol{b}) := \langle\phi(\boldsymbol{a}), \phi(\boldsymbol{b})\rangle$, denote $\boldsymbol{K} := \boldsymbol{\phi}^T\boldsymbol{\phi}$. We have that $K^{(i,j)} = \langle\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)\rangle = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Going back to our prediction function we have that, $f(\boldsymbol{x}; \boldsymbol{w}_{LS-R}) = \langle\boldsymbol{w}_{LS-R}, \phi(\boldsymbol{x})\rangle = \langle\boldsymbol{\phi}(\boldsymbol{K} + \lambda I)^{-1}\boldsymbol{y}^T, \phi(\boldsymbol{x})\rangle = \langle(\boldsymbol{K} + \lambda\boldsymbol{I})^{-1}\boldsymbol{y}^T, \phi(\boldsymbol{x})^T\boldsymbol{\phi}\rangle$. Let's denote $\boldsymbol{k} := \phi(\boldsymbol{x})^T\boldsymbol{\phi}$, where $k^{(i)} = \langle\phi(\boldsymbol{x}), \phi(\boldsymbol{x}_i)\rangle = k(\boldsymbol{x}, \boldsymbol{x}_i)$. We can then rewrite our prediction functions as $f(\boldsymbol{x}; \boldsymbol{w}_{LS-R}) := \boldsymbol{k}(\boldsymbol{K} + \lambda\boldsymbol{I})^{-1}\boldsymbol{y}^T$, note that $\phi(\boldsymbol{x})$ only appears in inner products! [1]

All we need now is a function, k, that mimics the behavior of the inner product (without actually having to compute the inner product) and if our $k(\boldsymbol{a}, \boldsymbol{b})$ is positive definite $\exists\phi$ such that $k(\boldsymbol{a}, \boldsymbol{b}) = \langle\phi(\boldsymbol{a}), \phi(\boldsymbol{b})\rangle$. Such a function is called a:

Definition 2.4 ***Kernel function:***
Function $k(\boldsymbol{a}, \boldsymbol{b})$ that mimics the inner product, if explicit $\phi(\boldsymbol{x})$ can be derived from k we say k induces feature transform $\phi(\boldsymbol{x})$. Many known choices of k exist.

2.3 Choosing a kernel function, k

First we give some examples of kernel functions,

Definition 2.5 ***Linear Kernel Function:***
$k(\boldsymbol{a}, \boldsymbol{b}) := \langle\boldsymbol{a}, \boldsymbol{b}\rangle$, *induces feature transform* $\phi(\boldsymbol{x}) = \boldsymbol{x}$

Definition 2.6 ***Polynomial Kernel Function with Degree b:***
$k(\boldsymbol{a}, \boldsymbol{b}) := (\langle\boldsymbol{a}, \boldsymbol{b}\rangle + 1)^b$, *this induces feature transform* $\phi(\boldsymbol{x}) = [\boldsymbol{x}, ..., \boldsymbol{x}^b]$.

Definition 2.7 ***RBF (or Gaussian) Kernel*** [2] ***:***
$k(\boldsymbol{a}, \boldsymbol{b}) := \exp(-\frac{||\boldsymbol{a}-\boldsymbol{b}||^2}{2\sigma^2})$, *induces feature transform $\phi(\boldsymbol{x})$ that is infinite-dimensional, σ is chosen before fitting (often as the median of the pairwise distances of all inputs).*

How do we choose an appropriate kernel function? it depends on the learning task and dataset. The RBF kernel is a good all-round choice for $\boldsymbol{x} \in \mathbb{R}^d$.

3 Probabilistic Model Selection in Regression

In frequentist model selection we wanted to minimize the expected squared error, however, this couldn't be done in practice, so we used an approximation called the out-sample error (definition 1.3). This approach, however, requires us to hold out part of our sample during training meaning:

1. We loose information (the data held out for testing).
2. Cross-validation helps combat the above point, however, leads to heavy calculation.
3. Our dataset may not be iid.

What about a probabilistic approach? In probabilistic model selection we model the uncertainty of a model, $m \in \{m_1, ..., m_k\}$, using a prior, $p(m)$, over the possible models. Then we can write the posterior of the model using Bayes rule: $p(m|D) \propto p(D|m)p(m)$. This expresses the preference over all models, $\{m_1, ..., m_k\}$, given the data, D.

But now that we have a model of preference over all models, how do we select a model? As Bayesians always do we marginalize! Our prediction function is $p(\hat{y}|D) = \sum_{m\in\{m_1,...,m_k\}} p(\hat{y}|D, m) \cdot p(m|D)$, which is a weighted sum of the prediction functions for each model, where the weights

[1] $\boldsymbol{K} = O(n^2)$ and $(\boldsymbol{K} + \lambda\boldsymbol{I})^{-1}$ is usually $O(n^3)$, so computational cost gets very large for a large n
[2] Note: RBF basis function and the RBF kernel are **not** the same thing.

are the model posteriors. Note that using Bayes rule we can write: $p(m|D) \propto p(D|m)p(m)$, where $p(D|m)$ is called the **Model Evidence**.

Now, how do we calculate $p(m|D)$? We have the model priors so we need to calculate $p(D|m)$. Suppose our model is parameterized by $\boldsymbol{w}$, then, $p(D|m) = \int p(D|\boldsymbol{w}, m) \cdot p(\boldsymbol{w}|m) d\boldsymbol{w}$. Note that we can write,

$$p(\boldsymbol{w}|D, m) = \frac{p(D|\boldsymbol{w}, m)p(\boldsymbol{w}|m)}{p(D|m)} \tag{11}$$

We notice that the model evidence ($p(D|m)$) normalizes the posterior of the parameters ($p(\boldsymbol{w}|D, m)$).

Is there a way of approximating the model evidence? Let's consider the case where we have only one parameter and suppose that $p(w|D, m) \propto p(D|w, m) \cdot p(w|m)$ plateaus at w_{MAP}. Then, $\int p(D|w, m) \cdot p(w|m) dw \approx p(D|w_{MAP}, m) \cdot p(w_{MAP}|m) \cdot \Delta posterior$ [3]. If the prior of the parameters is flat as well: $p(w|m) = \frac{1}{\Delta prior}$, then,

$$p(D|m) \approx p(D|w_{MAP}, m) \frac{\Delta posterior}{\Delta prior} \tag{12}$$

To further analyze the above lets consider its log, $\log p(D|m) \approx \log p(D|w_{MAP}, m) + \log \frac{\Delta posterior}{\Delta prior}$. Note that $\Delta posterior / \Delta prior < 1$ [4] therefore the second term is always negative and it increases in magnitude as the ratio gets smaller. So if parameters are finely tuned to the data in the posterior distribution the second term (the **penalty term**) gets large. The first term corresponds to the log likelihood (because the prior is flat).

Furthermore, for a model with b parameters we have: $\log p(D|m) \approx \log p(D|\boldsymbol{w}_{MAP}, m) + b \log \frac{\Delta posterior}{\Delta prior}$, assuming $\Delta posterior / \Delta prior$ the same for all w_i and that the w_i are independent (proof in section A.3). If we increase b the second term increases in magnitude (gets more negative), however, the first term will increase as a more complex model can better fit the data. So we have a trade-off where the model-evidence is optimal for some intermediate model-complexity.

3.1 Tuning hyper-parameters

In most scenarios we normally have that before trying to fit a model we must first parameterize the model such as choosing the regularization parameter or the degree (b) of the polynomial transform, these parameters are called **hyper parameters**. Can we use probabilistic model selection to help us determine a hyper-parameter? With hyper parameters we write the predictive distribution as:

$$p(\hat{y}|D) = \int p(\hat{y}|D, \alpha) \cdot p(\alpha|D) d\alpha = \int \int p(\hat{y}|\boldsymbol{w}\alpha) \cdot p(\boldsymbol{w}|D, \alpha) \cdot p(\alpha|D) \, d\boldsymbol{w} \, d\alpha \tag{13}$$

However, the integral wrt. α is intractable. If $p(\alpha|D)$ is very sharp and centred on $\hat{\alpha}$ then $\int \int p(\hat{y}|\boldsymbol{w}\alpha) \cdot p(\boldsymbol{w}|D, \alpha) \cdot p(\alpha|D) \, d\boldsymbol{w} \, d\alpha \approx \int p(\hat{y}|\boldsymbol{w}, \hat{\alpha}) \cdot p(\boldsymbol{w}|D, \hat{\alpha}) d\boldsymbol{w}$. However, to find $\hat{\alpha}$ we need to maximize $p(\alpha|D)$. Note that $p(\alpha|D) \propto p(D|\alpha)p(\alpha) = p(\alpha) \int p(D|\boldsymbol{w}, \alpha)p(\boldsymbol{w}|\alpha) d\boldsymbol{w}$. If $p(\alpha)$ is relatively flat then we just set $\hat{\alpha} := \text{argmax}_\alpha \int p(D|\boldsymbol{w}, \alpha) \cdot p(\boldsymbol{w}|\alpha) d\boldsymbol{w}$, this is called **Marginalized Likelihood Maximization** or **Evidence Approximation**.

[3] As $\int f(x)dx \approx f(x_0) \cdot \Delta x$, if f can be approximated by a step function with length Δx that peaks at x_0.

[4] As the posterior is almost always sharper than the prior.

Appendices

A Proofs

A.1 Proof of Bias-Variance Decomposition

By our data-generating assumption:

$$\begin{aligned}
\mathbb{E}_D([y_i - f(\boldsymbol{x}_i; \boldsymbol{w}_{LS})]^2 | \boldsymbol{x}_i) &= \mathbb{E}_D([g_{\boldsymbol{x}_i} + \epsilon_i - f(\boldsymbol{x}_i; \boldsymbol{w}_{LS})]^2 | \boldsymbol{x}_i) && (14)\\
&= \mathbb{E}_D([g_{\boldsymbol{x}_i} + \epsilon_i - f(\boldsymbol{x}_i; \boldsymbol{w}_{LS})]^2 + \mathbb{E}(f(\boldsymbol{x}_i; \boldsymbol{w}_{LS}) | \boldsymbol{x}_i) && (15)\\
&\quad - \mathbb{E}(f(\boldsymbol{x}_i; \boldsymbol{w}_{LS}) | \boldsymbol{x}_i)) && (16)\\
&= \mathbb{E}_D(\epsilon_i^2) + \mathbb{E}_D(\epsilon_i (g_{\boldsymbol{x}_i} - f(\boldsymbol{x}_i; \boldsymbol{w}_{LS})) + \mathbb{E}(f(\boldsymbol{x}_i; \boldsymbol{w}_{LS}) | \boldsymbol{x}_i) && (17)\\
&\quad - \mathbb{E}(f(\boldsymbol{x}_i; \boldsymbol{w}_{LS}) | \boldsymbol{x}_i)) + \mathbb{E}_D((g_{\boldsymbol{x}_i} - f(\boldsymbol{x}_i; \boldsymbol{w}_{LS}) && (18)\\
&\quad + \mathbb{E}(f(\boldsymbol{x}_i; \boldsymbol{w}_{LS}) | \boldsymbol{x}_i) - \mathbb{E}(f(\boldsymbol{x}_i; \boldsymbol{w}_{LS}) | \boldsymbol{x}_i))) && (19)\\
&= \mathbb{E}_D(\epsilon_i^2) + \mathbb{E}_D((g_{\boldsymbol{x}_i} - f(\boldsymbol{x}_i; \boldsymbol{w}_{LS}) + \mathbb{E}(f(\boldsymbol{x}_i; \boldsymbol{w}_{LS}) | \boldsymbol{x}_i) && (20)\\
&\quad - \mathbb{E}(f(\boldsymbol{x}_i; \boldsymbol{w}_{LS}) | \boldsymbol{x}_i))^2) && (21)\\
&= \mathrm{Var}(\epsilon_i) + \mathbb{E}_D((g_{\boldsymbol{x}_i} - \mathbb{E}(f(\boldsymbol{x}_i; \boldsymbol{w}_{LS}) | \boldsymbol{x}_i))^2) && (22)\\
&\quad + \mathbb{E}_D((\mathbb{E}(f(\boldsymbol{x}_i; \boldsymbol{w}_{LS}) | \boldsymbol{x}_i) - f(\boldsymbol{x}_i; \boldsymbol{w}_{LS}))^2) && (23)\\
&\quad + 2 \cdot \mathbb{E}_D((g_{\boldsymbol{x}_i} - \mathbb{E}(f(\boldsymbol{x}_i; \boldsymbol{w}_{LS}) | \boldsymbol{x}_i)) \cdot (\mathbb{E}(f(\boldsymbol{x}_i; \boldsymbol{w}_{LS}) | \boldsymbol{x}_i) && (24)\\
&\quad - f(\boldsymbol{x}_i; \boldsymbol{w}_{LS}))) && (25)
\end{aligned}$$

note that,

$$\mathbb{E}(f(\boldsymbol{x}_i; \boldsymbol{w}_{LS}) | \boldsymbol{x}_i) - f(\boldsymbol{x}_i; \boldsymbol{w}_{LS}) = f(\boldsymbol{x}_i; \boldsymbol{w}_{LS}) - f(\boldsymbol{x}_i; \boldsymbol{w}_{LS}) = 0 \tag{26}$$

so,

$$\mathbb{E}_D([y_i - f(\boldsymbol{x}_i; \boldsymbol{w}_{LS})]^2 | \boldsymbol{x}_i) = \mathrm{Var}(\epsilon_i) + (g_{\boldsymbol{x}_i} - \mathbb{E}(f(\boldsymbol{x}_i; \boldsymbol{w}_{LS}) | \boldsymbol{x}_i))^2 + \mathrm{Var}(f(\boldsymbol{x}_i; \boldsymbol{w}_{LS}) | \boldsymbol{x}_i) \tag{27}$$

hence, proven.

A.2 Proof of $\boldsymbol{w}_{LS-R} = \boldsymbol{\phi}(\boldsymbol{\phi}^T\boldsymbol{\phi} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}^T$

Let $\phi(\boldsymbol{X})$ be denoted by $\boldsymbol{\phi}$, recall that,

$$\boldsymbol{w}_{LS-R} := (\boldsymbol{\phi}\boldsymbol{\phi}^T + \lambda \boldsymbol{I})^{-1}\boldsymbol{\phi}\boldsymbol{y}^T \tag{28}$$

the Woodbury identity says,

$$(\boldsymbol{P}^{-1} + \boldsymbol{B}^T\boldsymbol{B})^{-1}\boldsymbol{B}^T = \boldsymbol{P}\boldsymbol{B}^T(\boldsymbol{B}\boldsymbol{P}\boldsymbol{B}^T + \boldsymbol{I})^{-1} \tag{29}$$

let $\boldsymbol{P} = \frac{1}{\lambda}\boldsymbol{I}$ and $\boldsymbol{B} = \boldsymbol{\phi}^T$, then,

$$\begin{aligned}
(\boldsymbol{\phi}\boldsymbol{\phi}^T + \lambda \boldsymbol{I})^{-1}\boldsymbol{\phi}\boldsymbol{y}^T &= \frac{1}{\lambda}\boldsymbol{I}\boldsymbol{\phi}(\boldsymbol{\phi}^T\frac{1}{\lambda}\boldsymbol{I}\boldsymbol{\phi} + \boldsymbol{I})^{-1} && (30)\\
&= \boldsymbol{\phi}(\boldsymbol{\phi}^T\boldsymbol{\phi} + \lambda \boldsymbol{I})^{-1} && (31)
\end{aligned}$$

hence, proven.

A.3 Proof that $\log p(D|m) \approx \log p(D|\boldsymbol{w}_{MAP}, m) + b \log \frac{\Delta \boldsymbol{posterior}}{\Delta \boldsymbol{prior}}$, assuming that $\frac{\Delta \boldsymbol{posterior}}{\Delta \boldsymbol{prior}}$ same $\forall w_i$ and w_i independent

We have that,

$$p(D|m) = \int p(D|\boldsymbol{w}, m) \cdot p(\boldsymbol{w}|m) d\boldsymbol{w} \tag{32}$$

Note that for all i,

$$p(w^{(i)}|m) \approx \frac{\Delta posterior}{\Delta prior} \tag{33}$$

so we have,

$$p(\boldsymbol{w}|m) = \prod_i p(w_i|m) \approx (\frac{\Delta posterior}{\Delta prior})^b \tag{34}$$

so,

$$p(D|m) \approx p(D|\boldsymbol{w}_{MAP}, m) \cdot (\frac{\Delta posterior}{\Delta prior})^b \tag{35}$$

Taking the log,

$$\log p(D|m) \approx \log p(D|\boldsymbol{w}_{MAP}, m) + b \log \frac{\Delta posterior}{\Delta prior} \tag{36}$$

hence, proven.

B Homeworks

B.1 For Section 1

Question B.1.1 ***Prove the bias-variance decomposition (definition 1.1):***
Proof in section A.1

B.2 For Section 2

Question B.2.1 ***Prove that*** $\boldsymbol{w}_{LS-R} = \boldsymbol{\phi}(\boldsymbol{\phi}^T\boldsymbol{\phi} + \lambda\boldsymbol{I})^{-1}\boldsymbol{y}^T$***:***
Proof in section A.2

B.3 For Section 3

Question B.3.1 ***Prove that*** $\log p(D|m) \approx \log p(D|\boldsymbol{w}_{MAP}, m) + b\log\frac{\Delta \boldsymbol{posterior}}{\Delta \boldsymbol{prior}}$ **,** ***assuming*** $\Delta\boldsymbol{posterior}/\Delta\boldsymbol{prior}$ ***the same for all*** w_i ***and that the*** w_i ***are independent.***
Proof in section A.3.

Question B.3.2 ***Example of using marginalized likelihood maximization/ evidence approximation in linear regression:***
Suppose we have a likelihood model:

$$p(\boldsymbol{y}_1 \cdots \boldsymbol{y}_n|\boldsymbol{x}_1 \cdots \boldsymbol{x}_n; \boldsymbol{w}, b) := \prod_{i\in D} N_{y_i}(\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i)\rangle, \sigma^2) \tag{37}$$

$$p(\boldsymbol{w}; \sigma_w, b) := N_{\boldsymbol{w}}(\boldsymbol{0}, \sigma_w^2\boldsymbol{I}) \tag{38}$$

we would like to find an expression for:

$$p(\boldsymbol{y}_1 \cdots \boldsymbol{y}_n|\boldsymbol{x}_1 \cdots \boldsymbol{x}_n; b, \sigma, \sigma_{\boldsymbol{w}}) \tag{39}$$

We have that,

$$p(\boldsymbol{y}_1 \cdots \boldsymbol{y}_n|\boldsymbol{x}_1 \cdots \boldsymbol{x}_n; b, \sigma, \sigma_{\boldsymbol{w}}) = \int p(\boldsymbol{y}_1 \cdots \boldsymbol{y}_n|\boldsymbol{x}_1 \cdots \boldsymbol{x}_n; \boldsymbol{w}, b, \sigma, \sigma_{\boldsymbol{w}}) \cdot p(\boldsymbol{w}) d\boldsymbol{w} \tag{40}$$

We can also rewrite,

$$\prod_{i\in D} N_{y_i}(\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i)\rangle, \sigma^2) = N_{\boldsymbol{y}}(\phi(\boldsymbol{X})^T\boldsymbol{w}, \sigma^2\boldsymbol{I}) \tag{41}$$

By 2.115 in PRML we have,

$$p(\boldsymbol{y}_1 \cdots \boldsymbol{y}_n|\boldsymbol{x}_1 \cdots \boldsymbol{x}_n; b, \sigma, \sigma_{\boldsymbol{w}}) = N_{\boldsymbol{y}}(\phi(\boldsymbol{X})^T\boldsymbol{0} + 0, \sigma^2\boldsymbol{I} + \phi(\boldsymbol{X})^T\sigma_{\boldsymbol{w}}^2\boldsymbol{I}\phi(\boldsymbol{X})) \tag{42}$$

$$= N_{\boldsymbol{y}}(\boldsymbol{0}, \sigma^2\boldsymbol{I} + \sigma_{\boldsymbol{w}}^2\phi(\boldsymbol{X})^T\phi(\boldsymbol{X})) \tag{43}$$