1 Bias-Variance Decomposition

We've previously studied the behaviour of least-square linear regression models of the form $f(\mathbf{x}; \mathbf{w}_{LS}) = \langle \boldsymbol{\phi}(\mathbf{x}_i), \mathbf{w}_{LS} \rangle$ on an i.i.d. dataset $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with $y_i = g(\mathbf{x}_i) + \epsilon_i$ for a feature transform $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^b$, an underlying function $g : \mathbb{R}^d \to \mathbb{R}$ and i.i.d. additive noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. One important result we noticed was that as the complexity of the model increased (through a larger b value and/or a smaller regularization term λ during computation of $\mathbf{w}_{LS}$) past a certain point the model started to perform worse on test data as it began **overfitting** to the training data—failing to generalise to unseen data as it placed to much importance on the particular noise within the training data. We have already used cross-validation as a tool to find an optimal model whose complexity was high enough to capture the behaviour of the underlying data-generating function g whilst not being too complex as to overfit to the training data, but a deeper frequentist analysis can help further illuminate the problem of overfitting.

If we assume each $\mathbf{x}_i$ is fixed, we can then define our testing error on a test set D_1 as $E(D_1, \mathbf{w}_{LS}) = \sum_{i \in D_1} [y_i - f(\mathbf{x}_i, \mathbf{w}_{LS})]^2$. Taking the expectation of this testing error over D (since we could choose any subset of D to be the test set) we see that

$$\mathbb{E}_D[E(D, \mathbf{w}_{LS})] = \mathbb{E}_D \left[\sum_{i \in D} [y_i - f(\mathbf{x}_i, \mathbf{w}_{LS})] \right] = \sum_{i \in D} \mathbb{E}_D \left[[y_i - f(\mathbf{x}_i, \mathbf{w}_{LS})]^2 | \mathbf{x}_i \right].$$

The expectation inside of the sum is the *expected loss*, and can be usefully decomposed into the following three terms (a proof of this can be found in Appendix A):

$$\mathbb{E}_D[[y_i - f_{LS}(\mathbf{x}_i)]^2 | \mathbf{x}_i] = \underbrace{\text{Var}[\epsilon]}_{\text{Irreducible Error}} + \underbrace{[g(\mathbf{x}_i) - \mathbb{E}[f_{LS}(\mathbf{x}_i) | \mathbf{x}_i]]^2}_{\text{Bias}^2} + \underbrace{\text{Var}[f_{LS}(\mathbf{x}_i) | \mathbf{x}_i]}_{\text{Variance}}.$$

The first term is beyond our control, it simply corresponds to the randomness inherent to the data-generating process we're trying to model. The second term (the bias) gives us the accuracy of our expected prediction—note that as we make the model more complex (e.g. by increasing b) the accuracy will increase and so this term will decrease. The third term (the variance) gives a a measure of how dependent the model is on the particular dataset (i.e. how much the dataset varies around its mean)—as the model complexity increases so too does this term.

Since we want a test error as small as possible, we want to minimize this expected loss, which will happen with a trade-off between decreasing the bias (through higher model complexity) and decreasing the variance (through lower model complexity). This helps to explain the overfitting behaviour we have seen previously, since overfitting to the training data corresponds to an increased variance when our model gets too complex.

1.1 In- & Out-Sample Error

We can extend this analysis by defining the *in-sample error* and *out-sample error*. The in-sample error is defined as the sum of the expected error on all of the $\mathbf{x}_i$ in our training set: $\frac{1}{n} \sum_{i=1}^n \mathbb{E}[(y_i - f_{LS}(\mathbf{x}_i))^2 | \mathbf{x}_i]$. Unfortunately, in practice we don't know g or the distribution of ϵ so these expectations can't be calculated. Instead we will use the out-sample error, taking the error over the distribution of $\mathbf{x}$ (which we no longer consider fixed), and denoting our training and testing sets as D_0 and D_1 respectively:

$$\begin{aligned} \mathbb{E}_{\mathbf{x}} \mathbb{E}[(y - f_{LS}(\mathbf{x}))^2 | \mathbf{x}] &= \mathbb{E}_{\mathbf{x}} \mathbb{E}_{D_1} \mathbb{E}_{D_0}[(y - f_{LS}(\mathbf{x}))^2 | \mathbf{x}] \\ &= \mathbb{E}_{p(\mathbf{x})} \mathbb{E}_{p(y|\mathbf{x})} \mathbb{E}_{D_0}[(y - f_{LS}(\mathbf{x}))^2 | \mathbf{x}] \\ &= \mathbb{E}_{D_0} \mathbb{E}_{p(y,\mathbf{x})}[(y - f_{LS}(\mathbf{x}))^2 | \mathbf{x}]. \end{aligned} \tag{1}$$

Again this can't be calculated directly as we don't know $p(y, \mathbf{x})$ but we can approximate it by considering multiple datasets $D^{(1)}, D^{(2)}, ..., D^{(K)}$ each split into a training set $D_0^{(k)}$ and a testing set $D_1^{(k)}$, containing pairs $(\mathbf{x}, y)$ from $p(\mathbf{x}, y)$. By writing $f_{LS}^{(k)}$ for the prediction function trained on $D_0^{(k)}$, we can then approximate 1 as

$$\mathbb{E}_{D_0}\mathbb{E}_{p(y,\mathbf{x})}[(y - f_{LS}(\mathbf{x}))^2|\mathbf{x}] \approx \frac{1}{K}\sum_{k=1}^{K}\frac{1}{|D_1^{(k)}|}\sum_{(\mathbf{x},y)\in D_1^{(k)}}\left(y - f_{LS}^{(k)}\right)^2.$$

Note that this if we were to let $D_0^{(k)}$ be the kth split of an i.i.d. dataset D and then let $D_1^{(k)} = D \setminus D_0^{(k)}$, the above result is essentially the same as K-fold cross-validation.

2 Feature Transform and Kernel Methods

Another way in which we can expand the least-squares analysis from previous weeks is by looking into our feature transforms $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^b$ in greater depth.

2.1 Feature Transforms

Previously we mainly considered the polynomial feature transform $\boldsymbol{\phi}(x) = [x, x^2, ..., x^b]^T$ (along with it's $b = 1$ version, the linear transform $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{x}^T$). The polynomial kernel tended to work well (with suitable b values) on smooth functions g, which can be justified by considering that smooth functions can be fairly well approximated by their Taylor series (e.g. around a point a and up to some order b by $g(x) \approx \sum_{i=0}^{b} g^{(i)}(a)(x-a)^i/(i!)$ where $g^{(i)}(a)$ is the ith derivative of g evaluated at a). However, there are many more choices that we can use for $\boldsymbol{\phi}$ by considering the different ways in which we can decompose a function g.

2.1.1 Trigonometric Transforms

A function can be written as its *Fourier series*: $g(x) = a_0 + \sum_{i=1}^{\infty}[a_i \sin(ix) + b_i(\cos(ix))]$ which is particularly useful if g is a periodic function (e.g. a sound wave or many other time-series). This leads us to the *trigonometric transform* in which the weights $\mathbf{w}$ are used as the a_i and b_i values in the Fourier series (note that here we have $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^{2b}$):

$$\boldsymbol{\phi}(x) := [\sin(x), \cos(x), \sin(2x), \cos(2x), ..., \sin(bx), \cos(bx)]^T.$$

2.1.2 Radial Basis Function

Both the polynomial and trigonometric transforms are based on approximating g by a *linear basis expansion* using the *basis functions* $\phi^{(i)}$: $g(\mathbf{x}) \approx f(\mathbf{x}; \mathbf{w}) = \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}) \rangle = \sum_{i=1} w^{(i)}\phi^{(i)}(\mathbf{x})$. Along with the polynomial basis and trigonometric basis, another very common choice for $\phi^{(i)}$ in regression problems is the *Radial Basis Function (RBF)* $\phi^{(i)}(\mathbf{x}) := \exp\left[-||\mathbf{x} - \mathbf{x}_i||^2/(2\sigma^2)\right]$ where $\sigma > 0$ is the *bandwidth* hyperparameter (chosen before fitting) and each $\mathbf{x}_i$ ($i \in [b]$) is called an RBF *centroid*. It is common to choose the centroids randomly from the dataset and set σ to be the median pairwise distance between all points $\mathbf{x}$ in the dataset.

With these basis functions, the feature transform $\boldsymbol{\phi}(\mathbf{x}) = [\phi^{(1)}(\mathbf{x}), ..., \phi^{(b)}(\mathbf{x})]^T$ then leads to weights $\mathbf{w}$ in which $w^{(i)}$ corresponds to how close $\mathbf{x}$ is to the centroid $\mathbf{x}_i$, increasing the value of $f(\mathbf{x}, \mathbf{w})$ if $w^{(i)} > 0$ and decreasing it if $w^{(i)} < 0$. It is clear then that a good RBF transform should use centroids from across the whole space of inputs $\mathbf{x}$—since although $\phi^{(i)}(\mathbf{x}) > 0$ for all $\mathbf{x}$, the further away $\mathbf{x}$ is from $\mathbf{x}_i$ the closer to zero it gets[1].

[1] And the rate at which it approaches zero is controlled by the bandwidth chosen beforehand.

It can be difficult to visualise the RBF feature transform in high-dimensional space (i.e. $\mathbb{R}^d$), but a general idea can be reached by considering b different d-dimensional balls of radius σ each centered around a centroid and thinking of $f(\mathbf{x};\mathbf{w})$ of being 'supported' in the regions of $\mathbb{R}^d$ in which there is a ball. (So by 'supported' we mean that at least one basis function $\phi^{(i)}$ is sufficiently large enough for $f(\mathbf{x};\mathbf{w})$ to not be approximately zero in this region.)

Clearly f can only be supported in all of $\mathbb{R}^d$ with an infinite number of centroids, however, we can cover a finite subset of this space—the subset in which we might actually expect an $\mathbf{x}$ to appear—with a finite number of centroids. However, this does raise the issue with the curse of dimensionality: the number of balls required to cover a space (the *packing number*) grows exponentially with the dimension of the space, which means so does the (ideal) number of centroids, something that should be taken into account when using RBFs.

2.2 Evaluating only the Inner Products

So far we've been using $\boldsymbol{\phi}$ to map our inputs from $\mathbb{R}^d$ to a b-dimensional feature space $\mathbb{R}^b$, but we might consider instead using an infinite-dimensional feature space. One problem that could arise, however, is in computing our LS solution $\mathbf{w}_{LS-R} = (\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^T + \lambda\mathbf{I})^{-1}\boldsymbol{\phi}(\mathbf{X})\mathbf{y}^T$ since this will be an infinitely long vector (to match the dimension of $\boldsymbol{\phi}(\mathbf{x})$) so in order to make computation tractable we'll have to rework our solution. The way in which we'll do this is by rewriting it as $\mathbf{w}_{LS-R} = (\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \lambda\mathbf{I})^{-1}\boldsymbol{\Phi}\mathbf{y}^T = \boldsymbol{\Phi}(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda\mathbf{I})^{-1}\mathbf{y}^T$ (where $\boldsymbol{\Phi} = \boldsymbol{\phi}(\mathbf{X})$). (A proof of this can be found in Appendix B.) This is much better since we will now calculate the $n \times n$ matrix $\boldsymbol{\Phi}^T\boldsymbol{\Phi} =: \mathbf{K}$ rather than $\boldsymbol{\Phi}\boldsymbol{\Phi}^T$, which is intractable.

Further, we define the function $k(\mathbf{x},\mathbf{y}) = \langle\boldsymbol{\phi}(\mathbf{x}),\boldsymbol{\phi}(\mathbf{y})\rangle$ which allows us to write each element of $\mathbf{K}$ as $K^{(i,j)} = \langle\boldsymbol{\phi}(\mathbf{x}_i),\boldsymbol{\phi}(\mathbf{x}_j)\rangle = k(\mathbf{x}_i,\mathbf{x}_j)$. Similarly, if we define $\mathbf{k} = \boldsymbol{\phi}(\mathbf{x})^T\boldsymbol{\Phi}$ (which has entries $k^{(i)} = \langle\boldsymbol{\phi}(\mathbf{x}),\boldsymbol{\phi}(\mathbf{x}_i)\rangle = k(\mathbf{x},\mathbf{x}_i)$), then our prediction function becomes:

$$\begin{aligned} f(\mathbf{x};\mathbf{w}_{LS-R}) = \langle\boldsymbol{\phi}(\mathbf{x}),\mathbf{w}_{LS-R}\rangle = \langle\boldsymbol{\phi}(\mathbf{x}),\boldsymbol{\Phi}(\mathbf{K}+\lambda\mathbf{I})^{-1}\mathbf{y}^T\rangle &= \langle\boldsymbol{\phi}(\mathbf{x})^T\boldsymbol{\Phi},(\mathbf{K}+\lambda\mathbf{I})^{-1}\mathbf{y}^T\rangle \\ &= \mathbf{k}(\mathbf{K}+\lambda\mathbf{I})^{-1}\mathbf{y}^T. \end{aligned}$$

In this form, we can compute $f(\mathbf{x};\mathbf{w}_{LS-R})$ without ever having to explicitly calculate $\boldsymbol{\phi}(\cdot)$ since it only ever appears inside an inner product function $k(\mathbf{x},\mathbf{y}) = \langle\boldsymbol{\phi}(\mathbf{x}),\boldsymbol{\phi}(\mathbf{y})\rangle$ (either in $\mathbf{k}$ or $\mathbf{K}$), therefore allowing us to use infinite-dimensional feature transforms through their corresponding inner product function k, which we refer to as a *kernel function*.

2.3 Kernel Function

To be able to use some kernel function $k(\cdot,\cdot)$ it must "behave like" an inner product. In particular, we can use the fact that if k is positive definite then there exists some $\boldsymbol{\phi}$ such that $k(\mathbf{x},\mathbf{y}) = \langle\boldsymbol{\phi}(\mathbf{x}),\boldsymbol{\phi}(\mathbf{y})\rangle$. If we can write out such a $\boldsymbol{\phi}(\mathbf{x})$ explicitly then we say that k *induces* the feature transform $\boldsymbol{\phi}(\mathbf{x})$. We could use a wide range of kernel functions, however, below we'll list a few common examples:

- **Linear kernel function**: $k(\mathbf{x},\mathbf{y}) = \langle\mathbf{x},\mathbf{y}\rangle$. This induces the linear transform $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{x}$.
- **Polynomial kernel function of degree b**: $k(\mathbf{x},\mathbf{y}) = (\langle\mathbf{x},\mathbf{y}\rangle + 1)^b$. With $b = 2$ this induces the linear feature transform

$$\begin{aligned} \boldsymbol{\phi}(\mathbf{x}) &= [(\mathbf{x}^{(1)})^2,...,(\mathbf{x}^{(d)})^2,\sqrt{2}\mathbf{x}^{(1)}\mathbf{x}^{(2)},...,\sqrt{2}\mathbf{x}^{(1)}\mathbf{x}^{(d)},\sqrt{2}\mathbf{x}^{(2)}\mathbf{x}^{(3)},...,\sqrt{2}\mathbf{x}^{(d-1)}\mathbf{x}^{(d)}, \\ &\qquad \sqrt{2}\mathbf{x}^{(1)},...,\sqrt{2}\mathbf{x}^{(d)},1]^T \\ &= [\forall_{k\in[d]}(\mathbf{x}^{(k)})^2,\forall_{1\le k<l\le d}\sqrt{2}\mathbf{x}^{(k)}\mathbf{x}^{(l)},\forall_{k\in[d]}\sqrt{2}\mathbf{x}^{(k)},1]^T \end{aligned}$$

  where $\mathbf{x}^{(k)}$ denotes the kth entry of $\mathbf{x} \in \mathbb{R}^d$. (See Appendix C for a proof of this.) Polynomial (and linear) kernels are often used in natural language processing.

- **RBF/Gaussian kernel**: $k(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{||\mathbf{x}-\mathbf{y}||^2}{2\sigma^2}\right)$ where σ is chosen before fitting (often as the median of pairwise distances between inputs $\mathbf{x}$). This induces an infinite-dimensional kernel that generally performs well when $\mathbf{x} \in \mathbb{R}^d$.

One problem with this kernel LS approach is that computational cost of calculating $(\mathbf{K} + \lambda\mathbf{I})^{-1}$ is $O(n^3)$, compared to the $O(b^3)$ cost of our previous (non-kernel-based) method. However, kernel methods are very flexible thanks to the wide range of kernels available—kernels can even be defined for data such as strings or graphs.

3 Probabilistic Model Selection

In section 1 we provided a frequentist analysis of the LS problem whereby we'd select the model with the lowest expected squared error. In practice, however, since g and σ aren't known, we have to instead use an approximation of the out-sample error through cross-validation, which is computationally expensive, excludes data in training to be used later for testing and relies on the i.i.d. assumption on our dataset. We might gain more insight by taking a probabilistic approach to model selection. To this end, we'll first choose some prior distribution on the models $p(m)$, which will allow us (via Bayes' rule) to find the posterior $p(m|D) \propto p(D|m)p(m)$. We can then define our predictive distribution as a marginalisation (i.e. a weighted sum of each model's prediction): $p(\hat{y}|D) = \sum_m p(\hat{y}|D, m)p(m|D)$. For this to work we must find a way to approximate the model evidence $p(D|m)$ (so that we can use $p(m|D)$ in the sum above).

Supposing the model m is governed by some parameters $\mathbf{w}$ we can write the model evidence as $p(D|m) = \int p(D|\mathbf{w}, m)p(\mathbf{w}|m)d\mathbf{w}$. Furthermore since $p(D|\mathbf{w}, m)p(\mathbf{w}|m) \propto p(\mathbf{w}|D, m)$, we make the assumption that $p(\mathbf{w}|D, m)$ can be approximated by a step function of 'width' $\Delta posterior > 0$ centered at $\mathbf{w}_{MAP}$. This then allows us to write

$$p(D|m) = \int p(D|\mathbf{w}, m)p(\mathbf{w}|m)d\mathbf{w} \approx p(D|\mathbf{w}_{MAP}, m)p(\mathbf{w}_{MAP}|m)\Delta posterior. \tag{2}$$

Furthermore, assuming that we can similarly write that $p(\mathbf{w}|m) = \frac{1}{\Delta prior}$ (i.e. $p(\mathbf{w}|m)$ is also flat with width $\Delta prior > 0$) then 2 becomes: $p(D|m) \approx p(D|\mathbf{w}_{MAP}, m)\frac{\Delta posterior}{\Delta prior}$. Taking the log of this we get $\log p(D|m) \approx \log p(D|\mathbf{w}_{MAP}, m) + \log \frac{\Delta posterior}{\Delta prior}$, the second term of which we note is negative (since posteriors are almost always sharper than priors so $\Delta posterior < \Delta prior$). Therefore we have a trade-off between the two terms in this expression. Assuming that a model has b parameters, all independent and with the same $\frac{\Delta posterior}{\Delta prior}$, this expression becomes $\log p(D|m) \approx \log p(D|\mathbf{w}_{MAP}, m) + b \log \frac{\Delta posterior}{\Delta prior}$ (a result which is proved in Appendix D). Hence, increasing b decreases $b \log \frac{\Delta posterior}{\Delta prior}$ (whilst the increased model flexibility increases $\log p(D|\mathbf{w}_{MAP}, m)$).

3.1 Tuning Hyper Parameters

We can use a similar approach to tune hyperparameters α for our predictive distribution $p(\hat{y}|D) = \int p(\hat{y}|D, \alpha)p(\alpha|D)d\alpha = \int\int p(\hat{y}|\mathbf{w}, \alpha)p(\mathbf{w}|D, \alpha)p(\alpha|D)d\mathbf{w}d\alpha$. This integral may be intractable, however, if $p(\alpha|D)$ is very concentrated at some $\hat{\alpha}$, we can evaluate the integral w.r.t. α at $\hat{\alpha}$, hence our predictive distribution becomes $\int\int p(\hat{y}|\mathbf{w}, \alpha)p(\mathbf{w}|D, \alpha)p(\alpha|D)d\mathbf{w}d\alpha \approx \int p(\hat{y}|\mathbf{w}, \hat{\alpha})p(\mathbf{w}|D, \hat{\alpha})d\mathbf{w}$, which is (hopefully) much easier to calculate.

To find $\hat{\alpha}$ we maximize $p(\alpha|D) \propto p(D|\alpha)p(\alpha) = p(\alpha)\int p(D|\mathbf{w}, \alpha)p(\mathbf{w}|\alpha)d\mathbf{w}$. In particular, if $p(\alpha)$ is relatively flat we may simply calculate $\hat{\alpha} := \text{argmax}_\alpha \int p(D|\mathbf{w}, \alpha)p(\mathbf{w}|\alpha)d\mathbf{w}$. This is known as "Marginalised Likelihood Maximisation" or "Evidence Approximation".

[Appendix E contains an example of probabilistic model selection for linear regression.]

A Bias-Variance Decomposition

A proof that

$$\mathbb{E}_D[[y_i - f_{LS}(\mathbf{x}_i)]^2|\mathbf{x}_i] = \text{Var}[\epsilon] + [g(\mathbf{x}_i) - \mathbb{E}[f_{LS}(\mathbf{x}_i)|\mathbf{x}_i]]^2 + \text{Var}[f_{LS}(\mathbf{x}_i)|\mathbf{x}_i].$$

Proof. First let us expand the LHS and use the fact that $y_i = g(\mathbf{x}_i) + \epsilon_i$ to obtain:

$$\begin{aligned}
\mathbb{E}_D[[y_i - f_{LS}(\mathbf{x}_i)]^2|\mathbf{x}_i] =& \mathbb{E}_D[y_i^2 - 2y_i f_{LS}(\mathbf{x}_i) + f_{LS}(\mathbf{x}_i)^2|\mathbf{x}_i] \\
=& \mathbb{E}_D[g(\mathbf{x}_i)^2 + 2\epsilon_i g(\mathbf{x}_i) + \epsilon_i^2 - 2(g(\mathbf{x}_i) + \epsilon_i)f_{LS}(\mathbf{x}_i) + f_{LS}(\mathbf{x}_i)^2|\mathbf{x}_i] \\
=& \mathbb{E}_D[\epsilon_i^2] + \mathbb{E}_D[g(\mathbf{x}_i)^2 - 2g(\mathbf{x}_i)f_{LS}(\mathbf{x}_i) + f_{LS}(\mathbf{x}_i)^2|\mathbf{x}_i] \\
&+ \mathbb{E}_D[2\epsilon_i(g(\mathbf{x}_i) - f_{LS}(\mathbf{x}_i))|\mathbf{x}_i] \\
=& \text{Var}[\epsilon_i] + \mathbb{E}[\epsilon_i]^2 + \mathbb{E}_D[g(\mathbf{x}_i)^2 - 2g(\mathbf{x}_i)f_{LS}(\mathbf{x}_i) + f_{LS}(\mathbf{x}_i)^2|\mathbf{x}_i] \\
&+ 2\mathbb{E}[\epsilon_i]\mathbb{E}_D[g(\mathbf{x}_i) - f_{LS}(\mathbf{x}_i)|\mathbf{x}_i] \\
=& \text{Var}[\epsilon] + 0 + \mathbb{E}_D[g(\mathbf{x}_i)^2 - 2g(\mathbf{x}_i)f_{LS}(\mathbf{x}_i) + f_{LS}(\mathbf{x}_i)^2|\mathbf{x}_i] + 0
\end{aligned}$$

in which we've used the fact that $\mathbb{E}[\epsilon_i] = 0$ and that ϵ_i is independent from $(g(\mathbf{x}_i) - f_{LS}(\mathbf{x}_i))$ when i is indexing the test set. Next we add $\mathbb{E}_D[f_{LS}(\mathbf{x}_i)|\mathbf{x}_i]^2 - \mathbb{E}_D[f_{LS}(\mathbf{x}_i)|\mathbf{x}_i]^2 = 0$ inside the second non-zero term:

$$\begin{aligned}
\mathbb{E}_D[[y_i - f_{LS}(\mathbf{x}_i)]^2|\mathbf{x}_i] =& \text{Var}[\epsilon] + \mathbb{E}_D[g(\mathbf{x}_i)^2 - 2g(\mathbf{x}_i)f_{LS}(\mathbf{x}_i) + \mathbb{E}_D[f_{LS}(\mathbf{x}_i)|\mathbf{x}_i]^2 \\
&+ \mathbb{E}_D[f_{LS}(\mathbf{x}_i)|\mathbf{x}_i]^2 + f_{LS}(\mathbf{x}_i)^2|\mathbf{x}_i] \\
=& \text{Var}[\epsilon] + \left(g(\mathbf{x}_i)^2 - 2g(\mathbf{x}_i)\mathbb{E}_D[f_{LS}(\mathbf{x}_i)|\mathbf{x}_i] + \mathbb{E}_D[f_{LS}(\mathbf{x}_i)|\mathbf{x}_i]^2\right) \\
&+ \left(\mathbb{E}_D[f_{LS}(\mathbf{x}_i)^2|\mathbf{x}_i] - \mathbb{E}_D[f_{LS}(\mathbf{x}_i)|\mathbf{x}_i]^2\right) \\
=& \text{Var}[\epsilon] + [g(\mathbf{x}_i) - \mathbb{E}[f_{LS}(\mathbf{x}_i)|\mathbf{x}_i]]^2 + \text{Var}[f_{LS}(\mathbf{x}_i)|\mathbf{x}_i].
\end{aligned}$$

□

B Rewriting The Least-Squares Solution

A proof that $\mathbf{w}_{LS-R} = (\mathbf{\Phi}\mathbf{\Phi}^T + \lambda\mathbf{I})^{-1}\mathbf{\Phi}\mathbf{y}^T = \mathbf{\Phi}(\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\mathbf{I})^{-1}\mathbf{y}^T$.

Proof. This is easily done by applying the Woodybury identity:

$$(P^{-1} + B^T B)^{-1}B^T = PB^T(BPB^T + \mathbf{I})^{-1}$$

with $P = \frac{1}{\lambda}\mathbf{I}$ (and hence $P^{-1} = \lambda\mathbf{I}$) and $B = \mathbf{\Phi}^T$, since

$$\begin{aligned}
\mathbf{w}_{LS-R} = (\mathbf{\Phi}\mathbf{\Phi}^T + \lambda\mathbf{I})^{-1}\mathbf{\Phi}^T\mathbf{y}^T &= (P^{-1} + B^T B)^{-1}B\mathbf{y}^T \\
&= PB^T(BPB^T + \mathbf{I})^{-1}\mathbf{y}^T \\
&= \frac{1}{\lambda}\mathbf{I}\mathbf{\Phi}\left(\mathbf{\Phi}^T\left(\frac{1}{\lambda}\mathbf{I}\right)\mathbf{\Phi} + \mathbf{I}\right)^{-1}\mathbf{y}^T \\
&= \mathbf{\Phi}(\lambda)^{-1}\left(\frac{1}{\lambda}\mathbf{\Phi}^T\mathbf{\Phi} + \mathbf{I}\right)^{-1}\mathbf{y}^T \\
&= \mathbf{\Phi}(\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\mathbf{I})^{-1}\mathbf{y}^T.
\end{aligned}$$

□

C Polynomial Kernel With $b = 2$

A proof that the polynomial kernel **b**: $k(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle + 1)^b$ with $b = 2$ induces the linear feature transform

$$\begin{aligned}\boldsymbol{\phi}(\mathbf{x}) &= [(\mathbf{x}^{(1)})^2, ..., (\mathbf{x}^{(d)})^2, \sqrt{2}\mathbf{x}^{(1)}\mathbf{x}^{(2)}, ..., \sqrt{2}\mathbf{x}^{(1)}\mathbf{x}^{(d)}, \sqrt{2}\mathbf{x}^{(2)}\mathbf{x}^{(3)}, ..., \sqrt{2}\mathbf{x}^{(d-1)}\mathbf{x}^{(d)}, \\ &\quad \sqrt{2}\mathbf{x}^{(1)}, ..., \sqrt{2}\mathbf{x}^{(d)}, 1]^T \\ &= [\forall_{k\in[d]}(\mathbf{x}^{(k)})^2, \forall_{1\le k<l\le d}\sqrt{2}\mathbf{x}^{(k)}\mathbf{x}^{(l)}, \forall_{k\in[d]}\sqrt{2}\mathbf{x}^{(k)}, 1]^T\end{aligned}$$

where $\mathbf{x}^{(k)}$ denotes the kth entry of $\mathbf{x} \in \mathbb{R}^d$.

Proof. Observe that

$$\begin{aligned}k(\mathbf{x}, \mathbf{y}) &= (\langle \mathbf{x}, \mathbf{y} \rangle + 1)^2 \\ &= \left(\sum_{k=1}^{d} x^{(k)}y^{(k)} + 1\right)^2 \\ &= \left[\sum_{k=1}^{d} x^{(k)}y^{(k)}\right]^2 + 2\sum_{k=1}^{d} x^{(k)}y^{(k)} + 1 \\ &= \left[\sum_{k=1}^{d}\sum_{l=1}^{d}(x^{(k)}x^{(l)})(y^{(k)}y^{(l)})\right] + \sum_{k=1}^{d}(\sqrt{2}x^{(k)})(\sqrt{2}y^{(k)}) + 1 \\ &= \left[\sum_{k=1}^{d}(x^{(k)})^2(y^{(k)})^2 + 2\sum_{k=1}^{d-1}\sum_{l=k+1}^{d}(x^{(k)}x^{(l)})(y^{(k)}y^{(l)})\right] + \sum_{k=1}^{d}(\sqrt{2}x^{(k)})(\sqrt{2}y^{(k)}) + 1 \\ &= \sum_{k=1}^{d}(x^{(k)})^2(y^{(k)})^2 + \sum_{k=1}^{d-1}\sum_{l=k+1}^{d}(\sqrt{2}x^{(k)}x^{(l)})(\sqrt{2}y^{(k)}y^{(l)}) + \sum_{k=1}^{d}(\sqrt{2}x^{(k)})(\sqrt{2}y^{(k)}) + 1 \\ &= \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{y}) \rangle\end{aligned}$$

where

$$\begin{aligned}\boldsymbol{\phi}(\mathbf{x}) &= [(\mathbf{x}^{(1)})^2, ..., (\mathbf{x}^{(d)})^2, \sqrt{2}\mathbf{x}^{(1)}\mathbf{x}^{(2)}, ..., \sqrt{2}\mathbf{x}^{(1)}\mathbf{x}^{(d)}, \sqrt{2}\mathbf{x}^{(2)}\mathbf{x}^{(3)}, ..., \sqrt{2}\mathbf{x}^{(d-1)}\mathbf{x}^{(d)}, \\ &\quad \sqrt{2}\mathbf{x}^{(1)}, ..., \sqrt{2}\mathbf{x}^{(d)}, 1]^T \\ &= [\forall_{k\in[d]}(\mathbf{x}^{(k)})^2, \forall_{1\le k<l\le d}\sqrt{2}\mathbf{x}^{(k)}\mathbf{x}^{(l)}, \forall_{k\in[d]}\sqrt{2}\mathbf{x}^{(k)}, 1]^T.\end{aligned}$$

□

D Model Evidence Approximation With b Parameters

A proof that a model with b parameters, each independent and with prior and posterior distributions that can be approximated by step functions of width $\Delta prior$ and $\Delta posterior$ respectively (in the fashion discussed in section 3) such that $\frac{\Delta posterior}{\Delta prior}$ is the same for each parameter, we then have:

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

Proof. Suppose that $\mathbf{w} = [w_1, ..., w_b]^T$ with each w_i independent (as we've already assumed) and let $\mathbf{w}^{(i)} := [w_1, ..., w_i]^T$ for $i \in [b]$. We saw before that with $b = 1$

$$\begin{aligned} p(D|m) = \int p(D|w_1, m)p(w_1|m)dw_1 &\approx p(D|w_{1_{MAP}}, m)p(w_{1_{MAP}}|m)\Delta posterior \\ &\approx p(D|w_{1_{MAP}}, m)\frac{\Delta posterior}{\Delta prior}. \end{aligned}$$

Extending this to a general $b \in \mathbb{N}$ we see that (thanks to the independence of each w_i)

$$\begin{aligned} p(D|m) &= \int p(D|\mathbf{w}, m)p(\mathbf{w}|m)d\mathbf{w} \\ &\approx p(D|w_{b_{MAP}}, m)\frac{\Delta posterior}{\Delta prior}\int p(D|\mathbf{w}^{(b-1)}, m)p(\mathbf{w}^{(b-1)}|m)d\mathbf{w}^{(b-1)} \\ &\approx p(D|w_{b_{MAP}}, m)p(D|w_{b-1_{MAP}}, m)\left(\frac{\Delta posterior}{\Delta prior}\right)^2\int p(D|\mathbf{w}^{(b-2)}, m)p(\mathbf{w}^{(b-2)}|m)d\mathbf{w}^{(b-2)} \\ &\vdots \\ &\approx \prod_{i=1}^{b} p(D|w_{i_{MAP}}, m)\frac{\Delta posterior}{\Delta prior} \\ &= p(D|\mathbf{w}_{MAP}, m)\left(\frac{\Delta posterior}{\Delta prior}\right)^b. \end{aligned}$$

Then taking the log of this expression we arrive at the desired result:

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

□

E Marginalised Likelihood Proof For Linear Regression

In a linear regression setting with

$$\begin{aligned} p(\mathbf{y}_1, ..., \mathbf{y}_n|\mathbf{x}_1, ..., \mathbf{x}_n; \mathbf{w}, b) &:= \prod_{i\in D}\mathcal{N}_{\mathbf{y}_i}(\langle\mathbf{w}, \boldsymbol{\phi}_b(\mathbf{x}_i)\rangle, \sigma^2\mathbf{I}) \\ &= \mathcal{N}_{\mathbf{y}}(\boldsymbol{\Phi}^T\mathbf{w}, \sigma^2\mathbf{I}) \end{aligned} \tag{3}$$

and

$$p(\mathbf{w}; \sigma_{\mathbf{w}}, b) := \mathcal{N}(\mathbf{0}, \sigma_{\mathbf{w}}^2\mathbf{I}_b) \tag{4}$$

we may calculate the marginalised likelihood as

$$\begin{aligned} p(\mathbf{y}_1, ..., \mathbf{y}_n|\mathbf{x}_1, ..., \mathbf{x}_n; b, \sigma, \sigma_{\mathbf{w}}) &= \int p(\mathbf{y}_1, ..., \mathbf{y}_n|\mathbf{x}_1, ..., \mathbf{x}_n; \mathbf{w}, b, \sigma, \sigma_{\mathbf{w}})p(\mathbf{w})d\mathbf{w} \\ &= \mathcal{N}_{\mathbf{y}}(\mathbf{0}, \sigma_{\mathbf{w}}^2\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \sigma^2\mathbf{I}). \end{aligned}$$

To see this, we'll use the Gaussian identity given by Pattern Recognition & Machine Learning (PRML) [1] equations 2.115-2.117. These say that, given a marginal Gaussian distribution for $\hat{\mathbf{x}}$ and a conditional Gaussian distribution for $\hat{\mathbf{y}}$ of the form:

$$p(\hat{\mathbf{x}}) = \mathcal{N}_{\hat{\mathbf{x}}}(\mu, \Lambda^{-1}) \tag{5}$$

$$p(\hat{\mathbf{y}}|\hat{\mathbf{x}}) = \mathcal{N}_{\hat{\mathbf{y}}}(\mathbf{A}\hat{\mathbf{x}} + \mathbf{b}, \mathbf{L}^{-1}), \tag{6}$$

the marginal distribution of $\hat{\mathbf{y}}$ is given by:

$$p(\hat{\mathbf{y}}) = \mathcal{N}_{\hat{\mathbf{y}}}(\mathbf{A}\mu, \mathbf{L}^{-1} + \mathbf{A}\Lambda^{-1}\mathbf{A}^T). \tag{7}$$

Note that equation 4 corresponds to equation 5 for $\hat{\mathbf{x}} = \mathbf{w}$ with $\mu = \mathbf{0}$ and $\Lambda^{-1} = \sigma_{\mathbf{w}}^2\mathbf{I}_b$. Furthermore, by setting $\mathbf{L}^{-1} = \sigma^2\mathbf{I}$, $\mathbf{A} = \mathbf{\Phi}^T$ and $\mathbf{b} = \mathbf{0}$ we can see how equation 3 corresponds to equation 6 for $\hat{\mathbf{y}} = \mathbf{y}$, thus allowing us to obtain the desired result from equation 7:

$$\begin{aligned} p(\mathbf{y}_1, ..., \mathbf{y}_n|\mathbf{x}_1, ..., \mathbf{x}_n; b, \sigma, \sigma_{\mathbf{w}}) &= \mathcal{N}_{\mathbf{y}}(\mathbf{\Phi}^T\mathbf{0}, \sigma^2\mathbf{I} + \mathbf{\Phi}^T(\sigma_{\mathbf{w}}^2\mathbf{I})\mathbf{\Phi}) \\ &= \mathcal{N}_{\mathbf{y}}(\mathbf{0}, \sigma_{\mathbf{w}}^2\mathbf{\Phi}^T\mathbf{\Phi} + \sigma^2\mathbf{I}). \end{aligned}$$

References

[1] Christopher M. Bishop. *Pattern recognition and machine learning*. Information science and statistics. Springer, New York, 2006.