1 Introduction: Statistical Decision Making

Good decision making requires that predictions be **precise** and **data-driven** whilst taking the **cost** of wrong predictions and the **random nature of data** into consideration. Historically this has been very difficult (e.g. augury and oracles), but thanks to the development of statistics and data science we now have the decision-making machinery to better follow these guidelines.

2 Least Squares Regression

We shall consider a regression problem with n observed pairs of d-dimensional real inputs $\mathbf{x} \in \mathbb{R}^d$ and one-dimensional real outputs $y \in \mathbb{R}$ collected in a dataset $D := \{(\mathbf{x}_i, y_i)\}_{i=1}^n$. The problem is then as follows: given a new input $\mathbf{x}^*$, predict its corresponding output y^*. More generally, the problem asks: can we find a prediction function $f(\mathbf{x})$ that gives us optimal predictions based on our dataset?

Least squares regression attempts this by finding a function $f : \mathbb{R}^d \to \mathbb{R}$ that minimizes the sum of squared errors $\sum_{i \in D_0} [y_i - f(\mathbf{x_i})]^2$ (here we are summing over a subset $D_0 \subseteq D$ of the dataset, called the *training set*).

2.1 Linear Least Squares

By writing our prediction function as $f(\mathbf{x}; \mathbf{w}) := \langle \mathbf{w}_1, \mathbf{x} \rangle + w_0$, where $\mathbf{w} = [\mathbf{w}_1, w_0]^T \in \mathbb{R}^{d+1}$ is a column vector of 'weights', we can optimize f via its parameter $\mathbf{w}$ to find the optimal least-squares weights $\mathbf{w}_{LS} = \text{argmin}_{\mathbf{w}} \sum_{i \in D_0} [y_i - f(\mathbf{x_i})]^2$.

If we consider each $\mathbf{x}_i$ as a column vector, let $\mathbf{y} = [y_1, ..., y_n] \in \mathbb{R}^n$ be a row vector, and construct our *design matrix* $\mathbf{X}$ as follows:

$$\mathbf{X} := \begin{bmatrix} \mathbf{x}_1 & ... & \mathbf{x}_n \\ 1 & ... & 1 \end{bmatrix}$$

then it is fairly easy to prove that $\mathbf{w}_{LS} = (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{y}^T$ (a proof is provided in Appendix A). Note that if $n < d$ then $\mathbf{X}\mathbf{X}^T$ is non-invertible (since $\mathbf{X}\mathbf{X}^T$ is not full-rank, being a $d \times d$ matrix with $\text{rank}(\mathbf{X}\mathbf{X}^T) \leq \text{rank}(\mathbf{X}) \leq \min(d, n) = n$), thus we need sufficient data ($n \geq d$) for this method to work.

This result is clearly precise and data-driven, however, if we want to take into consideration the randomness of the data we will have to examine the problem from a probabilistic point of view.

2.1.1 Maximum Likelihood Estimation (MLE)

Given an input $\mathbf{x}$ let us assume that $p(y|\mathbf{x}, \mathbf{w}, \sigma) = \mathcal{N}_y(f(\mathbf{x}; \mathbf{w}), \sigma^2)$, that is, y is normally distributed with mean $f(\mathbf{x}; \mathbf{w})$ and variance σ^2. If we further assume that each of the $(y_i, \mathbf{x_i})$ observations in our dataset are independent, we can write the likelihood of our observations as:

$$p(y_1, ..., y_n | \mathbf{x}_1, ..., \mathbf{x}_n, \mathbf{w}, \sigma) = \prod_{i=1}^n \mathcal{N}_{y_i}(f(\mathbf{x}_i; \mathbf{w}), \sigma^2).$$

Maximising this expression in order to find a maximum likelihood estimator for $\mathbf{w}$ (which we'll call $\mathbf{w}_{ML}$), we find that $\mathbf{w}_{ML} = \mathbf{w}_{LS}$. We can similarly find that the MLE for σ^2 is $\sigma^2_{ML} = \frac{1}{n} \sum_{i=1}^n (y_i - f(\mathbf{x}_i; \mathbf{w}_{ML}))^2$ (a proof for both of these results is provided in Appendix B).

2.1.2 Feature Transforms

We can easily extend this linear model to fit non-linear relationships by applying a feature transform $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^b$ to each input $\mathbf{x}$. For now we'll assume $d = 1$ and will use the polynomial transform $\boldsymbol{\phi}(x) = [x, x^2, x^3, ..., x^b]^T$ to construct our transformed design matrix as:

$$\boldsymbol{\phi}(\mathbf{X}) := \begin{bmatrix} \boldsymbol{\phi}(\mathbf{x}_1) & ... & \boldsymbol{\phi}(\mathbf{x}_n) \\ 1 & ... & 1 \end{bmatrix}$$

which, similarly to before, leads to the least-squares solution $\mathbf{w}_{LS} = (\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^T)^{-1}\boldsymbol{\phi}(\mathbf{X})\mathbf{y}^T$. (Furthermore, we show in Appendix C that if $\boldsymbol{\phi}(\mathbf{X})$ is symmetric and invertible then this simplifies to $\mathbf{w}_{LS} = [\boldsymbol{\phi}(\mathbf{X})]^{-1}\mathbf{y}^T$.)

We can easily extend this idea to datasets with $d > 1$, for instance if we denote the value of $\mathbf{x}$ in the jth dimension by $x^{(j)}$, and let $\mathbf{h}(t) = [t, t^2, t^3, ..., t^b]$, then we might consider the transform:

$$\boldsymbol{\phi}(\mathbf{x}) = [\mathbf{h}(x^{(1)}), \mathbf{h}(x^{(2)}), ..., \mathbf{h}(x^{(d)})]^T \in \mathbb{R}^{db}.$$

It is important to note that a wide range of feature transforms could be used, for instance the linear transform $\boldsymbol{\phi}(\mathbf{x}) = x$ would give us $\boldsymbol{\phi}(\mathbf{X}) = \mathbf{X}$, whilst other polynomial feature transforms could be constructed using cross-dimensional terms, such as adding all possible pairs and triplets of dimensions of $\mathbf{x}$ (along with a 1 to simplify construction of our design matrix):

$$\boldsymbol{\phi}(\mathbf{x}) = [1, \mathbf{h}(x^{(1)}), \mathbf{h}(x^{(2)}), ..., \mathbf{h}(x^{(d)}), \forall_{u<v} x^{(u)}x^{(v)}, \forall_{u<v<w} x^{(u)}x^{(v)}, x^{(v)}]^T.$$

However, since there are $\binom{d}{2}$ possible pairwise cross-dimension products and $\binom{d}{3}$ possible triple cross-dimension products, extending this out to the $\prod_u x^{(u)}$ term would lead to a design matrix with $\sum_{u=0}^{d} \binom{d}{u} = 2^d =: b$ rows (and n columns). This exponential growth in d could render the problem intractable due to storage and processing requirements—particularly since our method requires $n \geq b$ in order to work.

Problems like this (as well as others stemming from high-dimensional geometry) are often referred to as the 'curse of dimensionality'. Often we can perform dimensionality reduction on datasets (e.g. via principal component analysis) to represent them in fewer dimensions without losing too much information. In general it is important to note that using more dimensions in a feature transform doesn't necessarily lead to better results, in fact it can easily lead to overfitting.

2.2 Overfitting

A feature transform $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^b$ requires our model to learn the b parameters present in an optimal $\mathbf{w} \in \mathbb{R}^b$, hence we can think of b as a measure of our model's complexity. By analysing our model's performance on training and test sets we can see that as b increases we decrease the training error (as the added complexity allows the model to better fit the training data), however, the testing error decreases initially before starting to increase past certain values of b. This occurs because the more complex models start to fit to the noise of the training set, fitting less faithfully to the underlying data-producing process which the training and test set have in common. If a goal of regression is for the model to generalise well to unseen data then it is important that we avoid overfitting.

2.2.1 Cross-Validation

One very common technique to choose an optimal b that avoids overfitting is *cross-validation*, in which we split the overall dataset D into $k+1 \in \mathbb{N}$ disjoint subsets $D_0, ..., D_k$ and choose the value of b that minimizes the cross-validation error $\sum_i \text{testErr}(D_i)/(k+1)$ where testErr(D_i) gives us the test error of the model trained on $D \setminus D_i$ and tested on D_i.

Cross validation is, however, computationally expensive and depends on the hyperparameter k. It also relies on the IID assumption of our data, which would not be suitable for many datasets, such as time series.

2.3 Regularization

Another important way to deal with overfitting is *regularization*, wherein we attempt to keep the flexibility of high order polynomials whilst limiting the size of their coefficients in $\mathbf{w}$ in order to control this flexibility. This is done by introducing a value $\lambda > 0$ and including the regularization term $\lambda \mathbf{w}^T \mathbf{w}$ in our least-squares objective, at which point we see that

$$\begin{aligned}\mathbf{w}_{LS-R} :=& \underset{\mathbf{w}}{\text{argmin}} \sum_{i \in D_0} [y_i - f(\mathbf{x_i})]^2 + \lambda \mathbf{w}^T \mathbf{w} \\ =& (\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^T + \lambda \mathbf{I})^{-1} \boldsymbol{\phi}(\mathbf{X}) \mathbf{y}^T.\end{aligned}$$

(a proof for this can be found in Appendix D).

Note that as $\lambda \to \infty$ we find that $||\mathbf{w}_{LS-R}|| \to 0$ and thus $f(\mathbf{x}; \mathbf{w}_{LS-R}) \to 0$, i.e. larger values of λ flatten $f(\mathbf{x}; \mathbf{w}_{LS-R})$, reducing the model's complexity, which will (hopefully) lead to better generalization and avoid overfitting. Though of course we have to tune the value of λ as setting it too high can easily lead to underfitting. One way of analysing this is by returning to our probabilistic view of the problem.

2.3.1 Maximum A Posteriori (MAP)

Recall our likelihood function in which we assumed that each output y is independently normally distributed with mean $f(\mathbf{x}; \mathbf{w})$ and variance σ^2:

$$p(D|\mathbf{w}) = p(y_1, ..., y_n | \mathbf{x}_1, ..., \mathbf{x}_n, \mathbf{w}, \sigma) = \prod_{i=1}^{n} \mathcal{N}_{y_i}(f(\mathbf{x}_i; \mathbf{w}), \sigma^2).$$

Using Bayes' rule, the posterior distribution of $\mathbf{w}$ is $p(\mathbf{w}|D) = p(D|\mathbf{w})p(\mathbf{w})/p(D)$. If we choose a suitable prior for $\mathbf{w}$, say $p(\mathbf{w}) = \mathcal{N}_{\mathbf{w}}(0, \sigma_{\mathbf{w}}^2)$, we can then find the *maximum a posteriori* (MAP) estimator for $\mathbf{w}$ (noting that $p(D)$ does not depend on $\mathbf{w}$):

$$\begin{aligned}\mathbf{w}_{MAP} := \underset{\mathbf{w}}{\text{argmax}}\, p(\mathbf{w}|D) = \underset{\mathbf{w}}{\text{argmax}}\, p(D|\mathbf{w})p(\mathbf{w}) =& \underset{\mathbf{w}}{\text{argmax}} \prod_{i=1}^{n} \mathcal{N}_{y_i}(f(\mathbf{x}_i; \mathbf{w}), \sigma^2) \cdot \mathcal{N}_{\mathbf{w}}(0, \sigma_{\mathbf{w}}^2) \\ =& (\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^T + \frac{\sigma^2}{\sigma_{\mathbf{w}}^2} \mathbf{I})^{-1} \boldsymbol{\phi}(\mathbf{X}) \mathbf{y}^T.\end{aligned}$$

That is, $\mathbf{w}_{MAP} = \mathbf{w}_{LS-R}$ if we set $\lambda = \sigma^2/\sigma_{\mathbf{w}}^2$ (a proof of this can be found in Appendix E).

2.3.2 A Fully Probabilistic Approach

We can extend this probabilistic approach even further by calculating the predictive distribution $p(y^*|\mathbf{x}^*, D)$ of an output y^* given an input $\mathbf{x}^*$. Supposing that $f(\mathbf{x}^*; \mathbf{w}) = \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}^*) \rangle$ we then assume that $p(y^*|\mathbf{x}^*, \mathbf{w}) = \mathcal{N}_{y^*}(f(\mathbf{x}^*; \mathbf{w}), \sigma^2)$ which (through a proof found in Appendix F) leads us to the distribution:

$$p(y^*|\mathbf{x}^*, D) = \mathcal{N}_{y^*}\left[f(\mathbf{x}^*; \mathbf{w}_{MAP}), \sigma^2 + \boldsymbol{\phi}(\mathbf{x}^*)^T \sigma^2 \left(\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^T + \frac{\sigma^2}{\sigma_{\mathbf{w}}^2} \mathbf{I} \right)^{-1} \boldsymbol{\phi}(\mathbf{x}^*) \right].$$

This solution is useful as it not only gives us a prediction, but also tells us the uncertainty in that prediction, whilst being in a format that easily allows for more detailed investigations into the model's behaviour.

3 Binary Classification

So far we've dealt with regression, which predicts continuous values, however, it is important to also consider classification, in which we're making discrete decisions. In particular we'll look at binary classification: given and input $\mathbf{x} \in \mathbb{R}^d$ predict its output class $y \in \{+1, -1\}$. To do this, we search for a decision boundary $f(\mathbf{x}) = 0$ such that $f(\mathbf{x}) \leq 0$ gives us a prediction of $y = -1$ and $f(\mathbf{x}) > 0$ gives a prediction of $y = +1$. We can think of f splitting $\mathbb{R}^d$ into two regions: $R_- = \{\mathbf{x} : f(\mathbf{x}) \leq 0\}$ and $R_+ = \{\mathbf{x} : f(\mathbf{x}) > 0\}$.

3.1 Bayes Optimal Classifier

It makes sense that an optimal f would be found by minimizing the probability of misclassification:

$$\mathbb{P}(\mathbf{x} \text{ is misclassified}) = \int_{R_-} p(\mathbf{x}, y = +1) d\mathbf{x} + \int_{R_+} p(\mathbf{x}, y = -1) d\mathbf{x}$$

The solution to this is known as the **Bayes optimal classifier** and can be written simply as $f(\mathbf{x}) = p(\mathbf{x}, y = +1) - p(\mathbf{x}, y = -1)$ (a proof of this can be found in Appendix G). However, in practice we have to estimate $p(\mathbf{x}, y)$ as we don't have access to it directly.

3.2 Risk Minimisation

In many situations we may want to assign different costs to each type of misclassification, in which case we define a loss function L such that $L(y, y_0)$ tells us the cost of predicting a class y_0 when the true output is y. We can then optimize our classifier by minimizing the expected loss of making a wrong decision. Hence our classifier should make the decision given by

$$\underset{y_0}{\operatorname{argmin}} \, \mathbb{E}_{p(y|\mathbf{x})}[L(y, y_0)|\mathbf{x}].$$

Since we don't know $p(y|\mathbf{x})$, however, we need to replace it by $p(y|\mathbf{x}, D)$ and infer that from the dataset in some way. A *discriminative* approach to this would be to infer $p(y|\mathbf{x}, D)$ directly, perhaps using techniques such as MLE, MAP or a fully probabilistic method, however, we could instead use a *generative* approach in which we note that by Bayes' rule

$$p(y|\mathbf{x}, D) \propto p(\mathbf{x}|y, D)p(y).$$

Then (by similar methods) we can infer $p(\mathbf{x}|y, D)$ from the dataset, which not only allows us to predict output labels of inputs but also to generate new inputs given a class label (inference of $p(y)$ can simply be done through calculating the proportion of positive and negative samples in D).

One benefit of inferring $p(y|\mathbf{x})$ is being able to reject decision making altogether if we are not confident enough, e.g. if $\max(p(y = -1|\mathbf{x}), p(y = +1|\mathbf{x}) < \delta$ for some threshold $\delta > 0$. Having inferred $p(y|\mathbf{x})$ we can also provide a link to the regression problem—when the output y is continuous we can just use a loss function L such as squared loss (as in least-squares regression) $L(y, y_0) = (y - y_0)^2$ or absolute loss $L(y, y_0) = |y - y_0|$, and still the optimal prediction would be $\operatorname{argmin}_{y_0} \mathbb{E}_{p(y|\mathbf{x})}[L(y, y_0)|\mathbf{x}]$.

A Linear Least Squares

Proof that $\mathbf{w}_{LS} = (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{y}^T$:

$$\begin{aligned}\mathbf{w}_{LS} =& \underset{\mathbf{w}}{\text{argmin}} \sum_{i\in D_0} [y_i - f(\mathbf{x}_i)]^2 \\ =& \underset{\mathbf{w}}{\text{argmin}} \, ||\mathbf{y} - \mathbf{w}^T\mathbf{X}||_2^2 \\ =& \underset{\mathbf{w}}{\text{argmin}}\{(\mathbf{y} - \mathbf{w}^T\mathbf{X})(\mathbf{y} - \mathbf{w}^T\mathbf{X})^T\} \\ =& \underset{\mathbf{w}}{\text{argmin}}\{\mathbf{y}\mathbf{y}^T - \mathbf{y}\mathbf{X}^T\mathbf{w} - \mathbf{w}^T\mathbf{X}\mathbf{y}^T + \mathbf{w}^T\mathbf{X}\mathbf{X}^T\mathbf{w}\} \\ =& \underset{\mathbf{w}}{\text{argmin}}\{\mathbf{y}\mathbf{y}^T - 2\mathbf{y}\mathbf{X}^T\mathbf{w} + \mathbf{w}^T\mathbf{X}\mathbf{X}^T\mathbf{w}\}.\end{aligned}$$

Differentiating with respect to $\mathbf{w}$ and equating with zero we find that

$$\begin{aligned}0 =& -2\mathbf{X}\mathbf{y}^T + 2\mathbf{X}\mathbf{X}^T\mathbf{w}_{LS} \\ \mathbf{w}_{LS} =& (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{y}^T.\end{aligned}$$

B Maximum Likelihood Estimators

Proof that $\mathbf{w}_{ML} = \mathbf{w}_{LS}$ and $\sigma^2_{ML} = \frac{1}{n}\sum_{i=1}^n (y_i - f(\mathbf{x}_i; \mathbf{w}_{ML}))^2$:

Note that we actually maximise the log likelihood, which gives us the same MLEs since log is a monotonically increasing function. First to find $\mathbf{w}_{ML}$:

$$\begin{aligned}\mathbf{w}_{ML} :=& \underset{\mathbf{w}}{\text{argmax}} \prod_{i=1}^n \mathcal{N}_{y_i}(f(\mathbf{x}_i; \mathbf{w}), \sigma^2) \\ =& \underset{\mathbf{w}}{\text{argmax}} \log \prod_{i=1}^n \mathcal{N}_{y_i}(f(\mathbf{x}_i; \mathbf{w}), \sigma^2) \\ =& \underset{\mathbf{w}}{\text{argmax}} \left[\sum_{i=1}^n -\frac{(y_i - f(\mathbf{x}_i; \mathbf{w}))^2}{2\sigma^2}\right] - n\log\sigma\sqrt{2\pi} \\ =& \underset{\mathbf{w}}{\text{argmin}} \left[\sum_{i=1}^n (y_i - f(\mathbf{x}_i; \mathbf{w}))^2\right] \\ =& \mathbf{w}_{LS}.\end{aligned}$$

Secondly for σ^2_{ML}:

$$\begin{aligned}\sigma_{ML} :=& \underset{\sigma}{\text{argmax}} \prod_{i=1}^n \mathcal{N}_{y_i}(f(\mathbf{x}_i; \mathbf{w}), \sigma^2) \\ =& \underset{\sigma}{\text{argmax}} \left[\sum_{i=1}^n -\frac{(y_i - f(\mathbf{x}_i; \mathbf{w}))^2}{2\sigma^2}\right] - n\log\sigma\sqrt{2\pi} \\ =& \underset{\sigma}{\text{argmax}} -\frac{1}{2\sigma^2}\left[\sum_{i=1}^n (y_i - f(\mathbf{x}_i; \mathbf{w}))^2\right] - n\log(\sigma) - n\log(\sqrt{2\pi}).\end{aligned}$$

Differentiating this and equating it to zero we see that

$$\begin{aligned}
0 =& \frac{1}{\sigma_{ML}^3}\left[\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i;\mathbf{w}))^2\right] - \frac{n}{\sigma_{ML}} \\
n\sigma_{ML}^2 =& \left[\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i;\mathbf{w}))^2\right] \\
\sigma_{ML}^2 =& \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i;\mathbf{w}))^2.
\end{aligned}$$

C Feature-transformed Least Squares

Proof that if $\boldsymbol{\phi}(\mathbf{X})$ is symmetric and invertible then $\mathbf{w}_{LS} = [\boldsymbol{\phi}(\mathbf{X})]^{-1}\mathbf{y}^T$:

This can be seen from the previous expression for $\mathbf{w}_{LS}$ by applying the facts that (since $\boldsymbol{\phi}(\mathbf{X})$ is symmetric) $\boldsymbol{\phi}(\mathbf{X}) = \boldsymbol{\phi}(\mathbf{X})^T$, and also that $(AB)^{-1} = B^{-1}A^{-1}$ for two matrices A and B. Then,

$$\begin{aligned}
\mathbf{w}_{LS} &= [\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^T]^{-1}\boldsymbol{\phi}(\mathbf{X})\mathbf{y}^T \\
&= [\boldsymbol{\phi}(\mathbf{X})^T]^{-1}[\boldsymbol{\phi}(\mathbf{X})]^{-1}\boldsymbol{\phi}(\mathbf{X})\mathbf{y}^T \\
&= [\boldsymbol{\phi}(\mathbf{X})^T]^{-1}\mathbf{y}^T \\
&= [\boldsymbol{\phi}(\mathbf{X})]^{-1}\mathbf{y}^T.
\end{aligned}$$

D Regularized Least Squares

Proof that $\mathbf{w}_{LS-R} = (\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^T + \lambda\mathbf{I})^{-1}\boldsymbol{\phi}(\mathbf{X})\mathbf{y}^T$:

$$\begin{aligned}
\mathbf{w}_{LS-R} :=& \underset{\mathbf{w}}{\operatorname{argmin}}\{\sum_{i\in D_0}[y_i - f(\boldsymbol{\phi}(\mathbf{x}_i))]^2 + \lambda\mathbf{w}^T\mathbf{w}\} \\
=& \underset{\mathbf{w}}{\operatorname{argmin}}\{||\mathbf{y} - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{X})||_2^2 + \lambda\mathbf{w}^T\mathbf{w}\} \\
=& \underset{\mathbf{w}}{\operatorname{argmin}}\{(\mathbf{y} - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{X}))(\mathbf{y} - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{X}))^T + \lambda\mathbf{w}^T\mathbf{w}\} \\
=& \underset{\mathbf{w}}{\operatorname{argmin}}\{(\mathbf{y}\mathbf{y}^T - 2\mathbf{y}\boldsymbol{\phi}(\mathbf{X})^T\mathbf{w} + \mathbf{w}^T\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^T\mathbf{w} + \lambda\mathbf{w}^T\mathbf{w}.\}
\end{aligned}$$

Differentiating this with respect to $\mathbf{w}$ and equating it to zero we obtain

$$\begin{aligned}
0 =& -2\boldsymbol{\phi}(\mathbf{X})\mathbf{y}^T + 2\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^T\mathbf{w}_{LS-R} + 2\lambda\mathbf{w}_{LS-R} \\
(\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^T + \lambda\mathbf{I})\mathbf{w}_{LS-R} =& \boldsymbol{\phi}(\mathbf{X})\mathbf{y}^T \\
\mathbf{w}_{LS-R} =& (\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^T + \lambda\mathbf{I})^{-1}\boldsymbol{\phi}(\mathbf{X})\mathbf{y}^T.
\end{aligned}$$

E Maximum A Posteriori Least Squares

Proof that $\mathbf{w}_{MAP} = \mathbf{w}_{LS-R}$ with $\lambda = \sigma^2/\sigma_{\mathbf{w}}^2$:

$$\begin{aligned}
\mathbf{w}_{MAP} &:= \underset{\mathbf{w}}{\operatorname{argmax}}\, p(\mathbf{w}|D) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, p(D|\mathbf{w})p(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} \left[\prod_{i=1}^{n} \mathcal{N}_{y_i}(f(\mathbf{x}_i;\mathbf{w}),\sigma^2)\right] \cdot \mathcal{N}_{\mathbf{w}}(0,\sigma_{\mathbf{w}}^2) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} \left[\prod_{i=1}^{n} \exp\left(-\frac{1}{2\sigma^2}(y_i - f(\mathbf{x}_i;\mathbf{w}))^2\right)\right] \cdot \exp\left(-\frac{1}{2\sigma_{\mathbf{w}}^2}\mathbf{w}^T\mathbf{w}\right).
\end{aligned}$$

Taking the natural log this then becomes

$$\begin{aligned}
\mathbf{w}_{MAP} &= \underset{\mathbf{w}}{\operatorname{argmax}} \left[\sum_{i=1}^{n} -\frac{1}{2\sigma^2}(y_i - f(\mathbf{x}_i;\mathbf{w}))^2\right] - \frac{1}{2\sigma_{\mathbf{w}}^2}\mathbf{w}^T\mathbf{w} \\
&= \underset{\mathbf{w}}{\operatorname{argmin}} \left[\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i;\mathbf{w}))^2\right] + \frac{2\sigma^2}{2\sigma_{\mathbf{w}}^2}\mathbf{w}^T\mathbf{w} \\
&= \underset{\mathbf{w}}{\operatorname{argmin}}\, ||\mathbf{y} - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{X})||_2^2 + \frac{\sigma^2}{\sigma_{\mathbf{w}}^2}\mathbf{w}^T\mathbf{w} \\
&= \mathbf{w}_{LS-R} \quad (\text{with } \lambda = \frac{\sigma^2}{\sigma_{\mathbf{w}}^2}).
\end{aligned}$$

F Fully Probabilistic Predictive Distribution

Proof that $p(y^*|\mathbf{x}^*, D) = \mathcal{N}_{y^*}\left[f(\mathbf{x}^*;\mathbf{w}_{MAP}), \sigma^2 + \boldsymbol{\phi}(\mathbf{x}^*)^T\sigma^2\left(\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^T + \frac{\sigma^2}{\sigma_{\mathbf{w}}^2}\mathbf{I}\right)^{-1}\boldsymbol{\phi}(\mathbf{x}^*)\right]$:

We shall prove this by calculating $p(y^*|\mathbf{x}, D)$ as a marginalized distribution since

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

We have already assumed that $p(y^*|\mathbf{x}^*,\mathbf{w}) = \mathcal{N}_{y^*}(f(\mathbf{x}^*;\mathbf{w}),\sigma^2)$ so next we'll find the distribution $p(\mathbf{w}|D)$. We can do this up to a constant factor of $1/p(D)$ using Bayes' rule since

$$p(\mathbf{w}|D) \propto p(D|\mathbf{w})p(\mathbf{w}) = \left[\prod_{i=1}^{n}\mathcal{N}_{y_i}(f(\mathbf{x}_i;\mathbf{w}),\sigma^2)\right] \cdot \mathcal{N}_{\mathbf{w}}(0,\sigma_{\mathbf{w}}^2).$$

In a similar fashion to the previous proof, taking the natural log of this gives us

$$\begin{aligned}
\log p(\mathbf{w}|D) \propto &-\frac{1}{2\sigma^2}\left[\sum_{i=1}^{n} -(y_i - f(\mathbf{x}_i;\mathbf{w}))^2\right] - \frac{1}{2\sigma_{\mathbf{w}}^2}\mathbf{w}^T\mathbf{w} \\
= &-\frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{X}))(\mathbf{y} - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{X}))^T - \frac{1}{2\sigma_{\mathbf{w}}^2}\mathbf{w}^T\mathbf{w} \\
= &-\frac{1}{2\sigma^2}(\mathbf{y}\mathbf{y}^T - 2\mathbf{y}\boldsymbol{\phi}(\mathbf{X})^T\mathbf{w} + \mathbf{w}^T\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^T\mathbf{w}) - \frac{1}{2\sigma_{\mathbf{w}}^2}\mathbf{w}^T\mathbf{w} \\
= &-\frac{1}{2}\mathbf{w}^T\left[\frac{1}{\sigma^2}\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^T + \frac{1}{\sigma_{\mathbf{w}}^2}\mathbf{I}\right]\mathbf{w} + \frac{1}{\sigma^2}\mathbf{y}\boldsymbol{\phi}(\mathbf{X})^T\mathbf{w} + \text{const.}
\end{aligned}$$

Since $p(D|\mathbf{w})$ and $p(\mathbf{w})$ are both multi-variate Gaussian, $p(\mathbf{w}|D)$ will be too. Hence we can proceed by completing the square if we let $p(\mathbf{w}|D) = \mathcal{N}_{\mathbf{w}}(\mu,\Sigma)$ for some mean vector μ and covariance matrix Σ. Then

$$\log p(\mathbf{w}|D) = -\frac{1}{2}(\mathbf{w}-\mu)^T\Sigma^{-1}(\mathbf{w}-\mu) = -\frac{1}{2}\mathbf{w}^T\Sigma^{-1}\mathbf{w} + \mu^T\Sigma^{-1}\mathbf{w} + \text{const.}$$

Now by equating the quadratic coefficients of $\mathbf{w}$ in our two expressions for $\log p(\mathbf{w}|D)$ we can see that

$$\Sigma^{-1} = \frac{1}{\sigma^2}\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^T + \frac{1}{\sigma^2_{\mathbf{w}}}\mathbf{I}$$

and by equating the linear coefficients we similarly see that:

$$\begin{aligned}
\mu^T\Sigma^{-1} &= \frac{1}{\sigma^2}\mathbf{y}\boldsymbol{\phi}(\mathbf{X})^T \\
\mu^T &= \frac{1}{\sigma^2}\mathbf{y}\boldsymbol{\phi}(\mathbf{X})^T\left[\frac{1}{\sigma^2}\left(\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^T + \frac{\sigma^2}{\sigma^2_{\mathbf{w}}}\mathbf{I}\right)\right]^{-1} \\
\mu &= \frac{1}{\sigma^2}\left(\sigma^2\left[\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^T + \frac{\sigma^2}{\sigma^2_{\mathbf{w}}}\mathbf{I}\right]^{-1}\right)^T\boldsymbol{\phi}(\mathbf{X})\mathbf{y}^T \\
&= \left[\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^T + \frac{\sigma^2}{\sigma^2_{\mathbf{w}}}\mathbf{I}\right]^{-1}\boldsymbol{\phi}(\mathbf{X})\mathbf{y}^T \\
&= \mathbf{w}_{MAP}.
\end{aligned}$$

Now we have that

$$\begin{aligned}
p(\mathbf{w}|D) &= \mathcal{N}_{\mathbf{w}}(\mathbf{w}_{MAP}, \Sigma) \\
p(y^*|\mathbf{x}^*, \mathbf{w}) &= \mathcal{N}_{y^*}(f(\mathbf{x}^*; \mathbf{w}), \sigma^2) = \mathcal{N}_{y^*}(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}^*), \sigma^2),
\end{aligned}$$

so using a known result[1] for multi-variant normal distributions we can then obtain the desired result as

$$\begin{aligned}
p(y^*|\mathbf{x}^*, D) &= \mathcal{N}_{y^*}(\mathbf{w}^T_{MAP}\boldsymbol{\phi}(\mathbf{x}^*), \sigma^2 + \boldsymbol{\phi}(\mathbf{x}^*)\Sigma\boldsymbol{\phi}(\mathbf{x}^*) \\
&= \mathcal{N}_{y^*}\left[f(\mathbf{x}^*; \mathbf{w}_{MAP}), \sigma^2 + \boldsymbol{\phi}(\mathbf{x}^*)^T\sigma^2\left(\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^T + \frac{\sigma^2}{\sigma^2_{\mathbf{w}}}\mathbf{I}\right)^{-1}\boldsymbol{\phi}(\mathbf{x}^*)\right].
\end{aligned}$$

G Bayes Optimal Classifier

Proof that $f(\mathbf{x}) = p(\mathbf{x}, y = +1) - p(\mathbf{x}, y = -1)$ (the Bayes optimal classifier) minimizes the probability of misclassification

$$\mathbb{P}(\mathbf{x} \text{ is misclassified}) = \int_{R_-} p(\mathbf{x}, y = +1)d\mathbf{x} + \int_{R_+} p(\mathbf{x}, y = -1)d\mathbf{x} :$$

Using $\mathbf{1}$ as the indicator function, observe that

$$\begin{aligned}
f(\mathbf{x}) &= \operatorname*{argmin}_{f(\mathbf{x})}\left\{\int_{R_-} p(\mathbf{x}, y = +1)d\mathbf{x} + \int_{R_+} p(\mathbf{x}, y = -1)d\mathbf{x}\right\} \\
&= \operatorname*{argmin}_{f(\mathbf{x})}\left\{\int \left(p(\mathbf{x}, y = +1)\mathbf{1}_{\{f(\mathbf{x})\leq 0\}} + p(\mathbf{x}, y = -1)\mathbf{1}_{\{f(\mathbf{x})>0\}}\right)d\mathbf{x}\right\} \\
&= \operatorname*{argmin}_{f(\mathbf{x})}\left\{\int \left(p(\mathbf{x}, y = +1)(1 - \mathbf{1}_{\{f(\mathbf{x})>0\}}) + p(\mathbf{x}, y = -1)\mathbf{1}_{\{f(\mathbf{x})>0\}}\right)d\mathbf{x}\right\} \\
&= \operatorname*{argmin}_{f(\mathbf{x})}\left\{\int p(\mathbf{x}, y = +1)d\mathbf{x} + \int \left(p(\mathbf{x}, y = -1) - p(\mathbf{x}, y = +1)\right)\mathbf{1}_{\{f(\mathbf{x})>0\}}d\mathbf{x}\right\} \\
&= \operatorname*{argmin}_{f(\mathbf{x})}\left\{\int \left(p(\mathbf{x}, y = -1) - p(\mathbf{x}, y = +1)\right)\mathbf{1}_{\{f(\mathbf{x})>0\}}d\mathbf{x}\right\} \\
&= p(\mathbf{x}, y = +1) - p(\mathbf{x}, y = -1).
\end{aligned}$$

[1]See equation 2.115 in *Pattern Recognition and Machine Learning* by Christopher M. Bishop.