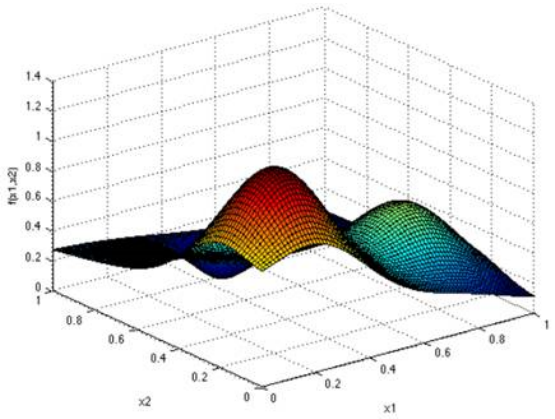

Figure 1: Shape of Franke's function which will be used as a interpolating goal. Note to self: Might be better to plot the graph instead of using a figure from the web. Remember to add the source for the figure by using for example bib. https://www.sfu.ca/ ssurjano/franke2d.html

1 Theory

1.1 Franke's function

Franke's function is a function which is often used to test different regression and interpolation methods. The function has two Gaussian peaks of differing heights, and a smaller dip. It's expression is

$$f(x,y) = \frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) + \frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}\right) + \frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) - \frac{1}{5}\exp\left(-(9x-4)^2 - (9y-7)^2\right).$$

and we define it for the interval $x, y \in [0, 1]$. An illustration of Franke's function can be seen in figure 1

1.2 Linear regression

The goal of a linear regression model is to find the coefficients $\hat{\beta}$ best suited for predicting new data via this expression:

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\beta},$$

where $\mathbf{X}$ is the design matrix constructed from the input data, and $\hat{\mathbf{y}}$ is the resulting prediction. To achieve this, we define a "cost function" of sorts, that evaluates each coefficients ability to predict the initial training dataset. We then

find the $\hat{\beta}$ that minimizes this cost function. More formally put, we want to find

$$\hat{\beta} = \arg\min_{\beta} C(\beta), \tag{1}$$

where $C(\beta)$ is the cost function.

1.2.1 Ordinary least squares regression (L_0)

Ordinary least squares (OLS) regression uses the residual sum of squares (RSS) function as the cost function. Given N datapoints and the predicted output $\mathbf{y}$, it reads

$$\text{RSS}(\beta) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2. \tag{2}$$

As found in appendix **??**, a cost function like (2) gives the following matrix equation for $\hat{\beta}$

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{3}$$

OLS regression has very low variance, but high bias - as a consequence of the bias-variance tradeoff. This makes OLS regression an "accurate" predictor of its own training data, but susceptible to overfitting, which the following two models are better suited to handle.

1.2.2 Lasso regression (L_1)

Lasso regression expands upon the above cost function by adding a term that penalizes the size of each coefficient. This is done by a factor of λ, as shown here:

$$\hat{\beta} = \arg\min_{\beta} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}. \tag{4}$$

The first sum is from the RSS function, while the second sum is the imposed penalty. The Lasso decreases bias from the OLS model, by decreasing the size of the coefficients, and thus making the variables more equally weighted.

Lasso regression has no closed form expression for $\hat{\beta}$, which means it must be calculated programatically.

1.2.3 Ridge regression (L_2)

Ridge regression has a penalty corresponding to the coefficient's squared sizes, further decreasing the bias from The Lasso, but obviously also increases variance. It defines $\hat{\beta}$ like this

$$\hat{\beta}^{\text{ridge}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\},$$

which - by the same proceedure as shown in appenix **??**, has this closed form expression

$$\hat{\beta}^{\text{ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}. \tag{5}$$