Answers: Exercise Sheet No. 11

1 Theory questions

1. (a) Maximum likelihood estimation is an approach to estimate parameters given some observed data and model. In particular, the objective is to maximize the likelihood that the data was generated by the model, i.e. find the parameters such that the observed data is most probable. When the data is assumed to be normally distributed then MLE is equivalent to least squares regression.

   (b) The likelihood function describes the likelihood of observing the data given a model and model parameters, i.e. $p(X|\theta)$

   (c) Although MLE can be equivalent to least squares estimation, there are choices of distributions that give a different objective (and hence a different estimate).

2. (a) One can compare the fitted model to *new* data. Also one can look at the sensitivity of the estimated parameters to changes in the data, create an uncertainty estimate of the parameters, etc.

   (b) Over-fitting can be addressed by regularisation, changing the objective function, or using a model with less degrees of freedom. Underfitting requires a model with more degrees of freedom.

3. (a) Over-fitting is when a model fits very well to the data used in the optimization but predicts new data very poorly (or at least much worse than the original data).

   (b) Regularisation penalizes the model parameters from taking non-zero values. Because of this it acts against the expresivity of the model, and promotes solutions that have some near-zero parameters.

   (c) $l1$ and $l2$ regression, (penalizing the absolute and squared values of the parameters respectively). Note that there are other options, but these are very common.

2 Ordinary Least squares

1. If you don't know that certain variables do not influence the fit then you would have 'too many' independent variables. E.g. a reaction rate is only influenced by compounds A and B, but you include compound C in the regression.

2. Code below.

3. Code below.

4. Code below.

5. Code below.

6. Code below. Both regularised solutions should have smaller values of the fitted parameters, especially the parameters that should not matter to the solution (the extra 7 parameters). The relatively large non-zero parameters in the least squares solution means that even though these parameter should not matter, the least squares solution is assigning importance to these variables.

7. When increasing the number of data-points the effect of overfitting decreases, and the influence of regularisation (with a fixed parameter value) decreases.

8. If the regularisation parameters become too large then the bias induced by regularisation dominates and you have a very poor fit as you are now underfitting.

9. The $l1$ norm gives a non-smooth problem which can sometimes be problematic for derivative based optimization algorithms.

```python
import numpy as np

# Generate random data
num_samples = 20
num_var = 10
X = np.random.rand(num_samples, num_var)

# Generate weights for features
weights = np.random.rand(3)
# Add some noise to the output
noise = np.random.normal(0, 0.1, num_samples)
# Calculate the expected output using the weights and input data
y = np.dot(X[:,0:3], weights) + noise

# Split data into training and testing sets
split_ratio = 0.5
num_train_samples = int(num_samples * split_ratio)
X_train = X[:num_train_samples]
X_test = X[num_train_samples:]
y_train = y[:num_train_samples]
y_test = y[num_train_samples:]


# Define the regularisation parameter
alpha = 0.01 # use this in your code for l2
beta = 0.0001 # use this for l1

# Fit a linear regression model using maximum likelihood estimation
theta = np.dot(np.linalg.inv(np.dot(X_train.T, X_train)), np.dot(
    X_train.T, y_train))



# Fit a linear regression model using L2 regularisation
theta_l2 = np.dot(np.linalg.inv(np.dot(X_train.T, X_train) + alpha
    * np.identity(num_var)), np.dot(X_train.T, y_train))


# Fit a linear regression model using L1 regularisation
def fun(theta):
    return sum( (y_train - np.dot(X_train,theta))**2) + beta*sum(
    abs(theta))
```

```
40
41 theta_l1  = sol = scipy.optimize.minimize(fun, np.zeros(
       num_features), method='Nelder-Mead').x
42
43
44
45 # Predict the output using the different models
46 y_pred = np.dot(X_test, theta)
47 y_pred_l1 = np.dot(X_test, theta_l1)
48 y_pred_l2 = np.dot(X_test, theta_l2)
49
50 # Calculate the mean squared error for the different models
51 mse = np.mean((y_test - y_pred) ** 2)
52 mse_l1 = np.mean((y_test - y_pred_l1) ** 2)
53 mse_l2 = np.mean((y_test - y_pred_l2) ** 2)
54
55 # Print the mean squared error for the different models
56 print("Mean Squared Error without regularisation: {:.4f}".format(
       mse))
57 print("Mean Squared Error with L1 regularisation: {:.4f}".format(
       mse_l1))
58 print("Mean Squared Error with L2 regularisation: {:.4f}".format(
       mse_l2))
```