5 Regression

In this question you will train models for regression and analyze a dataset. Start by downloading the code and dataset from the website.

The dataset is created from data provided by UNICEF's State of the World's Children 2013 report: `http://www.unicef.org/sowc2013/statistics.html`

Child mortality rates (number of children who die before age 5, per 1000 live births) for 195 countries, and a set of other indicators are included.

5.1 Getting started

Run the provided script `polynomial_regression.py` to load the dataset and names of countries / features.

Answer the following questions about the data. Include these answers in your report.

1. Which country had the highest child mortality rate in 1990? What was the rate?
2. Which country had the highest child mortality rate in 2011? What was the rate?
3. Some countries are missing some features (see original .xlsx/.csv spreadsheet). How is this handled in the function `assignment1.load_unicef_data()`?

For the rest of this question use the following data and splits for train/test and cross-validation.

- **Target value**: column 2 (Under-5 mortality rate (U5MR) 2011)[1].
- **Input features**: columns 8-40.
- **Training data**: countries 1-100 (Afghanistan to Luxembourg).
- **Testing data**: countries 101-195 (Madagascar to Zimbabwe).
- **Cross-validation**: subdivide training data into folds with countries 1-10 (Afghanistan to Austria), 11-20 (Azerbaijan to Bhutan), ... I.e. train on countries 11-100, validate on 1-10; train on 1-10 and 21-100, validate on 11-20, ...

5.2 Polynomial Regression

Implement linear basis function regression with polynomial basis functions. Use only monomials of a single variable (x_1, x_1^2, x_2^2) and no cross-terms ($x_1 \cdot x_2$).

Perform the following experiments:

1. Create a python script `polynomial_regression.py` for the following.

[1] Zero-indexing, hence `values[:,1]`.

Fit a polynomial basis function regression (unregularized) for degree 1 to degree 6 polynomials. Plot training error and test error (in RMS error) versus polynomial degree.

Put this plot in your report, along with a brief comment about what is "wrong" in your report.

Normalize the input features before using them (not the targets, just the inputs $\boldsymbol{x}$). Use `assignment1.normalize_data()`.

Run the code again, and put this new plot in your report.

2. Create a python script `polynomial_regression_1d.py` for the following.

   Perform regression using just a single input feature.

   Try features 8-15 (Total population - Low birthweight). For each (un-normalized) feature fit a degree 3 polynomial (unregularized).

   Plot training error and test error (in RMS error) for each of the 8 features. This should be a bar chart (e.g. use `matplotlib.pyplot.bar()`).

   Put this bar chart in your report.

   The testing error for feature 11 (GNI per capita) is very high. To see what happened, produce plots of the training data points, learned polynomial, and test data points. The code `visualize_1d.py` may be useful.

   In your report, include plots of the fits for degree 3 polynomials for features 11 (GNI), 12 (Life expectancy), 13 (literacy).

5.3 ReLU Basis Function

1. Create a python script `relu_regression.py` for the following.

   Implement regression using a modified version of ReLU basis function for a single input feature. Mathematically, ReLU is defined as $f(x) = max(0, x)$. For this part, use the modified ReLU defined as $f(x) = max(0, g(x))$, where $g(x) = -x + 5000$. Include a bias term. Use un-normalized features.

   Fit this regression model using feature 11 (GNI per capita).

   In your report, include a plot of the fit for feature 11 (GNI).

   In your report, include the training and testing error for this regression model.

5.4 Regularized Polynomial Regression

1. Create a python script `polynomial_regression_reg.py` for the following.

   Implement L_2-regularized regression. Fit a degree 2 polynomial using $\lambda = \{0, .01, .1, 1, 10, 10^2, 10^3, 10^4\}$. Use normalized features as input. Use 10-fold cross-validation to decide on the best value for λ. Produce a plot of average validation set error versus λ. Use a `matplotlib.pyplot.semilogx` plot, putting λ on a log scale[2].

[2]The unregularized result will not appear on this scale. You can either add it as a separate horizontal line as a baseline, or report this number separately.

Put this plot in your report, and note which λ value you would choose from the cross-validation.