ML Model Training

Peter Chapin

Vermont State University

CIS-2730, Software Engineering Projects

Goals

- Input Data
  - Temperature readings were taken at various times of day
  - The readings were taken at multiple locations in the Winooski Valley of Vermont.
- Predictive Output
  - The temperature at locations other than where the readings were taken, but also in the Winooski Valley (interpolation)
  - The temperature at times other than when the readings were taken (interpolation)
  - We are not interested in predicting temperatures outside the valley (extrapolation)

Approach

- We'll use a model with 9 tunable parameters.
- Divide input data into (training, validation & test) sets
  - We will normalize the input features to the range -1.0 to +1.0 (probably not necessary for longitude and latitude, but why not do it?)
  - We will *not* use separate validation and test data
- Adjust the parameters using gradient descent with the training data.
  - We will use an evaluation function of squared error (i.e., we will do a least-squares fit)
- Check the model against the validation & test data.

Normalization

- *Talk about normalization*

Training, Validation, Test

- *Talk about training, validation, and test data sets*

The Model

- Many models are possible.
  - Convolutional neural networks (CNNs) are good at dealing with spatially distributed information, such as recognizing pictures of cats.
  - Long-Short-Term Memory (LSTM) models are good at dealing with time-sequential data where it is necessary to “remember” information over long periods of time (e.g., seasonal variations).
  - ConvLSTM2D models are tuned for information that varies over 2D space and time. This might be a perfect fit for our situation.
  - We could set this up using TensorFlow and Keras (Python libraries for machine learning with neural networks). Anaconda comes with them!

What We'll Do

- We're going to do something simpler.
  - We know the temperature varies as `-cos(t)`; that is, it is cold at night and warm during the day.
  - We want to adjust the baseline, amplitude, and phase shift in a way that depends on location. Thus:

$$B(x, y) - A(x, y)\cos(t + P(x, y))$$

We’ll Use Planes

- For our purposes, we’ll use tilted planes for the spatial functions.
  - Here, *x* is (normalized) longitude, and *y* is (normalized) latitude.

$$B(x, y) = b_0 + b_1 x + b_2 y$$
$$A(x, y) = a_0 + a_1 x + a_2 y$$
$$P(x, y) = p_0 + p_1 x + p_2 y$$

  - The goal is to find values for the nine parameters that minimize the total squared error.
  - We can set $b_0 = 20.0$, $a_0 = 5.0$, and $p_0 = 0.0$ with all other values zero initially

Evaluation Function

- For a given set of tunable parameter values:
  - For each item in the training data set:
    - Compute the error between the predicted temperature and the “real” temperature.
    - Square the error and add it to an accumulated error.
  - Divide the accumulated error by the number of data points.
  - Take the square root of the result (root-mean-squared or *RMS* error).
- The goal is to adjust the parameters repeatedly to find the minimum RMS error.

Gradient Descent

- *Talk about gradient descent*