CP218: Theory and Applications of Bayesian Learning
Report on Kaggle project

Vilasini Ashokan
SR Number: 06-18-01-10-51-24034

1 Overview

In this project, a dataset containing temperature and humidity for 5 different locations calculated by 5 sensors at the sample rate of 10 seconds has been provided. The data corresponding to each sensor has 200 missing values. For each missing value row, either temperature or humidity is missing but not both. We are required to find the missing values using a probabilistic approach. I have tried two approaches: Bayesian regression and Gaussian Mixture Model.
Data preprocessing: The dataset contains errors in the year value. Instead of 2014 and 2015, '0014' and '0015' have been given, which were causing problems in the calculation of the Unix time. The reason for choosing Unix time is to keep a standard numerical value for the time, so that it becomes easier during model implementation of regression as well as GMM. For each node data, rows with missing values have been taken as test set and the others have been taken as train set. They have been scaled using StandardScaler function of scikit-learn as scaling handles features with different units and improves stability and convergence of the model.

2 Methodology

2.1 Model 1: Bayesian Linear Regression

Motivation: The most natural and simple choice for predicting missing values for any dataset is interpolation, which involves fitting a linear/polynomial model. Since we are required to approach this in a probabilistic way, Bayesian Linear regression is motivated as a strong starting point.
Mathematical formulation and assumptions: For each sensor node data, 'temperature' and 'humidity' are assumed to be a linear function of the 'Unix time'. Mathematically, Temperature $= w0 + w1 *$ Unix time and Humidity $= w2 + w3 *$ Unix time where the prior weights are assumed to be $\mathcal{N}(0, \lambda I)$. If U represents the vector of Unix time of the train data and T_{train} denotes the temperature vector of the train data, the posterior weights is given by $w_{post} = (U^t U + \lambda I)^{-1} U^t T_{train}$. Finally, the predicted value of temperature for test data is given by $T_{test} = U_{test} w_{post}$ where U_{test} denotes the Unix time vector of the test data points. Similar calculations hold for Humidity predicted values for the test data. In the code, I've calculated the linear regression between: (a) Temperature and Unix time, (b) Humidity and Unix time, (c) Temperature and Humidity(temp as function of humidity), and (d) Humidity and Temperature(humidity as function of temperature). In the test data, wherever temperature is missing, the average of prediction from Temperature-Unix time regression and Temperature-Humidity regression is taken. Similarly, wherever humidity is missing, the average of prediction from Humidity-Unix time regression and Humidity-temperature regression is considered.
Hyper-parameter selection: λ is chosen to be 1.
Advantages: 1. The λI term acts as a regularizer constant and it ensures that the model is not very sensitive to noise or small sample sizes.
2. It is a very simple model, has easy execution,less computational time and computationally stable.
3. Instead of relying on a single regression model, averaging across two models improves accuracy.
Disadvantages: 1. This has a very high RMSE value: 7.74 which clearly indicates the inefficiency to capture the true relation between the features.
2. This model doesn't take into account the location details and the temporal relations.

2.2 Model 2: Bayesian polynomial regression

Motivation: Since the Bayesian linear regression fails to capture the true complex relations between the features of the data, it is good to try polynomial regression of degree 15.
Mathematical formulation and assumptions: For each training example, if T is temperature, H is

humidity and t is Unix time, then it is assumed that T = $\beta_0+\beta_1 t+...+\beta_1 5t^{15}$ and H = $\delta_0+\delta_1 t+...+\delta_{15}t^{15}$ where vector $\beta \sim \mathcal{N}(0, \lambda_1^{-1}I)$ and vector $\delta \sim \mathcal{N}(0, \lambda_2^{-1}I)$ for some regularization parameters λ_1 and λ_2. The posterior weight for temperature is given by $U^tU + \lambda_1 I)^{-1}U^tT_{train}$ where $U = (1, t, t^2, ..., t^15)^t$ and T_{train} denotes the vector containing temperature of all the training exmaples. Similar calculation holds for humidity. Here also, average over predictions with respect to two observed features is taken for the missing feature.

Hyper parameter selection: $\lambda_1 = \lambda_2 = 2$ gives a better estimate than the linear regression. The RMSE value reduces to 6.53 for this model. Taking $\lambda_1 = \lambda_2 = 3$ gives RMSE value 6.56.

Advantages: 1. Higher order polynomial allows the non-linear relations to be captured in the value prediction.

2. The model has an analytical solution for the weights, which is computationally efficient for moderate-sized datasets.

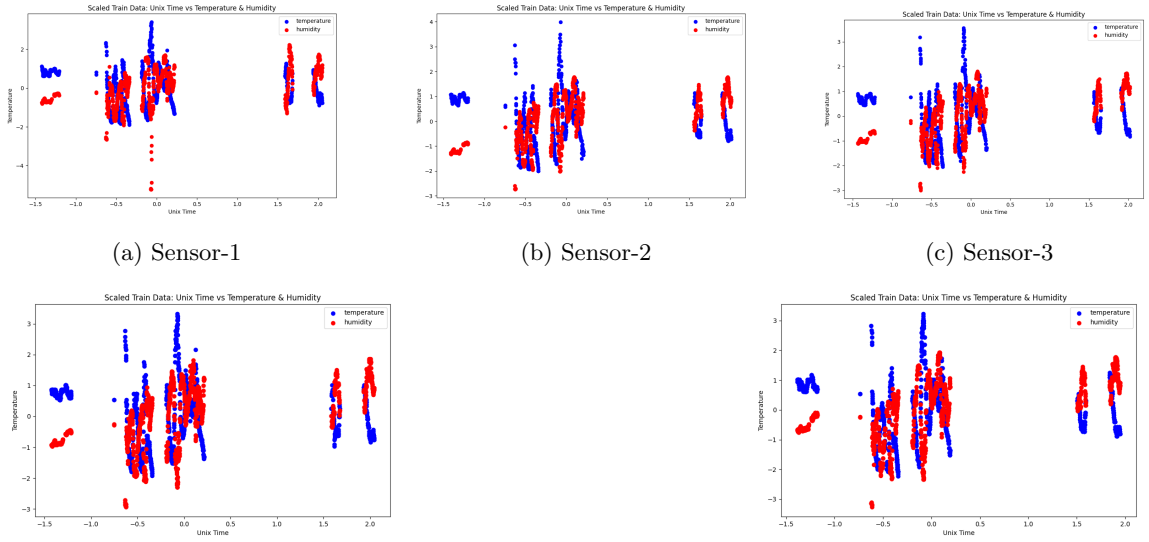
Disadvantages: 1. Even with regularization, 15th degree polynomial might overfit if the dataset is small or noisy.

2. Still inefficient in capturing the true complex relation between the features of the given data.

3. Computationally expensive compared to linear model. Not worthy for the small reduction in RMSE.

2.3 Model 3(Final Model): Gaussian Mixture Model

Motivation: The below graphs show relation between temperature(and humidity) as a function of Unix time for 5 sensor data.



(a) Sensor-1 (b) Sensor-2 (c) Sensor-3

(d) Sensor-4 (e) Sensor-5

Figure 1: Plots showing relation between temperature(and Humidity) and Unix time

The above plots suggest that the relation between Temperature(equi. Humidity) and Unix time is not linear, and not even polynomial and that is why previous models failed to effectively predict the values. In fact, the data forms clusters and each cluster has some probability distribution. This motivates us to try Gaussian Mixture Model. It uses distinct Normal distributions for different clusters which is aligning with our requirement.

Mathematical formulation and assumptions: It is assumed that each training(and test) example is independent. Let K be the number of clusters. If Z is the true cluster for a data point, then P(Z) follows Multinoulli(π). Let x_i be the i-th training example which is a 3 component vector containing Unix time, temperature and humidity values. Then by Bayes theorem,

$$\gamma_{ic}(\text{called responsibility}) = P(Z = c|x) = \frac{P(x|Z = c)P(Z = c)}{P(x)} = \frac{\pi_c \mathcal{N}(x_i|\mu_c, \Sigma_c)}{\sum_{i=1}^{K} \pi_i \mathcal{N}(x_i|\mu_i, \Sigma_i)}.$$

Since we do not know the closed form expression for P(Z), we use EM algorithm which evaluates the mean, variance and responsibilities iteratively using E-step and M-step. There are various initial values that could be considered; one e.g. is mean of each cluster=mean of the entire training data, variance=identity,

$\pi = (1/K, 1/K, 1/K, 1/K, 1/K)$. I used GaussianMixture built-in function of scikit-learn to fit training data using GMM model. Finally, the missing data is predicted using the formula:

$$x_{missing} = \sum_{j=1}^{K} (\gamma_{ij} * E[x_{missing}|x_{obs}, j])$$

Now, $E[x_{missing}|x_{obs}, k] = \mu_{missing} + \Sigma_{cross}.T(\Sigma_{obs})^{-1}(x_{obs} - \mu_{obs})$ where $\mu_{missing}$ denotes mean of the missing feature component, μ_{obs} denotes the mean of observed feature components, Σ_{obs} denotes the covariance matrix of observed feature components and Σ_{cross} denotes the covariance matrix of missing feature component with the observed feature components.
Hyperparameter selection: Below table suggests K=45 to be the best, but with compromise of computational cost.

Clusters	5	15	25	35	45
RMSE	3.629	2.566	2.205	2	1.92

Table 1: Number of clusters VS RMSE

Advantages: 1. Unlike K-Means, GMM can model elliptical soft clustering and multimodal distributions and captures complex relations through distributions among the features. Soft clustering useful when data points lie at cluster boundaries.
2. As the values are calculated using Gaussian distributions, the interpolation is very smooth. EM algorithm provides stable and accurate calculations.
3. GMM has the capability to model clusters having different densities, orientations, and sizes.
Disadvantages: 1. We require to fitting multiple Gaussian models. Inversion of multiple covariance matrices slows down the process.
2. If the data is not Gaussian, the model is inefficient.
3. It is sensitive to hyper-parameters and also the covariance matrix inversion becomes unstable if the feature space is very large.
4. This model doesn't capture the temporal and spatial dependencies and relations.
Why the model works: GMM models the joint distribution of temperature, humidity, and timestamp using multiple Gaussian components. This is an advantage over simple linear and polynomial regression, which assumes a single global relationship. The conditional expectation formula used ensures that missing values are estimated based on the strongest relationships in the data. Also the model is based on clustering which aligns with the original data pattern.
Other approaches: The biggest disadvantage of Gaussian Mixture Model is that it doesn't capture the spatial relation between the features as it doesn't make use of location data. Also, it requires large number of clusters(like 45) to give less RMSE value.
There are various models which accomplishes it such as:
1. **Gaussian Process Regression**: It is a non-parametric Bayesian approach which assumes that the function f from input to output is given by a Gaussian process with mean function μ and covariance function K(Kernel function). **Advantages:** 1. Due to sequential nature of sensor data, GPR can model temporal dependencies.
2. The flexible Kernel are effective in finding spatial information.
Why wasn't it used: GPR scales poorly with large datasets with $O(n^3)$ complexity for n data points. Given that each sensor node has thousands of timestamps, GPR will be very slow. Also, GPR works with one target variable at a time. Since we need to predict both temperature and humidity, GMM is a better choice as it models joint probability distributions more naturally.
2. **Bayesian Neural Network:** It extends traditional neural networks by placing probability distributions over weights instead of fixed values. Advantages of BNN: can capture nonlinear relationships and unlike GPR, BNN scales better with large datasets when implemented efficiently. **Why wasn't it used:** It is difficult to train and tune, can overfit to the training data is not regularized properly, and also very computationally expensive.

Learning: Understood the importance of joint probability modelling. I learnt that scaling and efficiency matter in real world data. That is why probabilistic models with EM and GMM algorithms are preferable due to moderate computational cost and effective results.

Possible improvements: Instead of fixing the number of clusters K manually, we could use a Bayesian Information Criterion (BIC) or Variational Bayesian GMM for hyperparameter tuning. Also, we can incorporate spatial dependencies (Kriging) into GMM for a spatiotemporal imputation model.

References:

1. Stanford CS229 lecture on GMM by Andrew NG.
2. NPTEL lecture on GMM by Prof. C.A. Murthy.
3. Geeksforgeeks notes on GPR.