CP2: Bayesian Linear Regression

Last modified: 2023-02-24 12:55

Status: **RELEASED.**

What to turn in:

- ZIP file of source code: https://www.gradescope.com/courses/496674/assignments/2698008/
- PDF report file: https://www.gradescope.com/courses/496674/assignments/2697971/

Your ZIP file should include

- All starter code .py files (with your edits) (in the top-level directory)

Your PDF should include (in order):

- Your full name
- Collaboration statement
- Problem 1 short answers
- Problem 2 figures
- Problem 3 figures

Advice: Problems 1 and 2 together are worth 90% of this CP. If you run low on time, omitting Problem 3 won't hurt you too badly.

Questions?: Post to the cp2 topic on the discussion forums.

Jump to: Problem 1 Problem 2 Problem 3 Starter Code

Goals

In CP2 we implement the Bayesian linear regression model discussed in class and in your readings.

The problems below will address two key questions:

- Given training data, how should we estimate the probability of a new observation? How might estimates change if we have very little (or abundant) training data?
- How can we select hyperparameter values to improve our predictions on heldout data? Can we contrast methods like cross validation with methods that might use our whole training set more effectively?

Probabilistic Model

We are given N data examples of input/output pairs: $\{x_n, t_n\}_{n=1}^N$.

We will form a joint distribution for linear regression tasks that models both the weights and the observed "outputs".

Prior model

We place a multivariate Normal prior over the weight vector $w \in \mathbb{R}^M$.

$$p(w) = \mathcal{N}(w|0, \alpha^{-1} I_M)$$

Here, the scalar $\alpha > 0$ is the "prior precision" hyperparameter.

Likelihood model

Next, we model the N observed outputs $t = \{t_n\}_{n=1}^N$ as follows:

$$p(t|x, w) = \prod_{n=1}^{N} \mathcal{N}(t_n|w^T \phi(x_n), \beta^{-1})$$

Here, the scalar $\beta > 0$ is the "likelihood precision" hyperparameter.

We'll assume the raw features x_n are just one dimensional, and we will use *polynomial* features for the feature transform:

$$\phi^{(0)}(x_n) = [1]^T$$
$$\phi^{(1)}(x_n) = [1\ x_n]^T$$
$$\phi^{(2)}(x_n) = [1\ x_n\ x_n^2]^T$$

where we get to pick the *order* r of the polynomial as a *hyperparameter*.

Key Concept 1: How should we estimate the probability of new outputs given new inputs?

We'll compare two ways to form a predictive distribution for new t_* given x_*

Maximum A-Posteriori (MAP) estimator

We can solve the following MAP estimation problem to get one estimate of vector w:

$$w_{\text{MAP}} = \arg\max_w \log p(w|\{x_n, t_n\}_{n=1}^N, \alpha, \beta)$$
$$= \left(\Phi^T\Phi + \frac{\alpha}{\beta}I\right)^{-1}\Phi^T t$$

See discussion around Eq. 3.52 - 3.58 for how to see why this estimate will solve the MAP estimation problem.

(Side note: Recall that we could also motivate this estimate not via prior probabilities, but by a view that just seeks a "penalized" ML estimator, combining the maximum likelihood objective with an additional term that penalizes the sum of squares of "w" with strength $\lambda > 0$. See Bishop's textbook Eq. 3.28 for the penalized ML estimator of w.)

Given the MAP estimate of w, when presented with a new input/output data pair t_*, x_*, we can compute its likelihood via the following univariate Normal PDF:

$$p(t_*|x_*, w = w_{\text{MAP}}, \beta) = \mathcal{N}(t_*|w_{\text{MAP}}^T\phi(x_*), \beta^{-1})$$

We get this directly from our i.i.d. likelihood assumptions, now applied to new data point $*$ rather than the original N data points.

Posterior Predictive estimator

The MAP estimator above requires us to commit to one particular weight vector. We might instead want to estimate the entire posterior *distribution*, and then average over this distribution when estimating what new outputs should look like.

This average-over-the-posterior strategy yields the Posterior Predictive distribution:

$$p(t_*|x_*, \{x_n, t_n\}_{n=1}^N, \alpha, \beta) = \mathcal{N}(t_*|m_N^T\phi(x_*), \sigma_N^2(x_*))$$

Where the mean m_N is from Bishop PRML Eq. 3.53 and the variance σ_N^2 is from Bishop PRML Eq. 3.59.

Key Concept 2: How to select the hyperparameter settings using training data?

We'll compare two strategies to select α, β:

Grid search with Cross Validation (see Problem 2)

Using cross validation, we can perform a grid search to select one best hyperparameter setting from a candidate set of values, as outlined in pseudocode below.

1. Divide dataset into K=5 folds
2. Compute score for each hyperparameter candidate
* fold1score = score on fold 5 after training on folds 1,2,3,4
* fold2score = score on fold 4 after training on folds 1,2,3,5
* fold3score = score on fold 3 after training on folds 1,2,4,5
* fold4score = score on fold 2 after training on folds 1,3,4,5
* fold5score = score on fold 1 after training on folds 2,3,4,5
* average_score = mean([fold1score, fold2score, fold3score, fold4score, fold5score])
3. Select hyperparameter with highest average_score

Grid Search with Evidence (see Problem 3)

Alternatively, using the evidence strategy we do:

1. For each candidate hyperparameter, compute evidence on train set (see textbook Eq. 3.86)
2. Select hyperparameter with best evidence

Data and Starter Code

You can find the starter code and dataset in the course Github repository here:

https://github.com/tufts-ml-courses/cs136-23s-assignments/tree/main/unit2_CP/

This starter code assumes you have activated the standard conda environment from this class (`spr_2021s_env`).

Inside the `data/` folder of the starter code repo, you'll find CSV files for a dataset called **toywave**. This is a *synthetic* dataset made by your course staff to be similar to "sine wave" data described in Bishop Fig. 1.4

```
Properties of toywave data
* 512 train examples (scalar feature-label pairs)
* 512 test examples (scalar feature-label pairs)
```

Problem 1

Tasks for Code Implementation

Steps (i) and (ii) will be evaluated by Autograder

CODE 1(i) Implement `fit` and `predict` and `predict_variance` methods of LinearRegressionMAPEstimator.py

CODE 1(ii) Implement `fit` and `predict` and `predict_variance` methods of LinearRegressionPosteriorPredictiveEstimator.py

CODE 1(iii) Edit `run_demo_MAP.py` and `run_demo_PPE.py` to solve the tasks below.

Keep `alpha = 0.01` fixed throughout Problem 1.

Tasks for Report PDF

1a: SHORT ANSWER : Given a dataset of size N, how do we score the model's predictions? Translate the provided starter code for the `score` function of the MAP estimator into a mathematical expression involving our probabilistic model. Provide a function in terms of parameters w_{MAP}, β and dataset $\{x_n, t_n\}_{n=1}^{N}$.

1b: SHORT ANSWER : Explain why different β values might be preferred by different model orders. (Hint: Execute `run_demo_MAP.py` at 3 different β values: {4, 100, 2500}. Look at the MAP estimator's scores on the training-set to see which β is preferred by which order. Then study the provided visualization to get intuition about why.).

1c: SHORT ANSWER : Compare the visuals produced by `run_demo_PPE.py` to those from the MAP estimator. Why might we prefer the posterior predictive estimator (PPE) to the MAP? (Hint: What happens to the predictions of flexible models at x values far from train data?).

Problem 2: Model Selection via heldout score averaged over 5-fold Cross-Validation

In problem 1, we set α and β manually. To do *better*, we'll now explore one principled way to select the value of α, β via grid search to optimize performance: K=5 fold cross validation.

We'll look at the following candidate hyperparameters in our *grid search*:

- 7 possible alpha parameters: [0.0001, 0.0010, 0.0100, 0.1000, 1.0000, 10.0000, 100.0000]
- 25 possible beta parameters: [0.010, 0.018, 0.032, 0.056, 0.100, 0.178, 0.316, 0.562, 1.000, 1.778, 3.162, 5.623, 10.000, 17.783, 31.623, 56.234, 100.000, 177.828, 316.228, 562.341, 1000.000, 1778.279, 3162.278, 5623.413, 10000.000]

Our goal is to determine the best values of α, β for each polynomial order in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9].

Tasks for Code Implementation

CODE 2(i) Implement the `main` method of `run_grid_search__5fold_heldout_score.py`, filling in required TODOs

Tasks for Report PDF

2a: FIGURE: Probabilistic predictions of CV-selected models

Show a figure that visualizes probabilistic predictions on N=20 training data for your MAP models that were selected to maximize the average score of 5-fold CV. Each panel shows the best model for a different polynomial feature transform {order 1, order 3, order 9}, using the alpha/beta values *selected specifically for that order*.

Each panel should show:

- solid line: predicted mean for t as a function of the provided inputs x
- filled area around line: +/-3 standard deviations, to understand estimator's uncertainty in predictions (using matplotlib's `fill_between`)

Use the starter code: most of this plot is already provided.

2b: FIGURE: Model Score vs. Polynomial Order

Your figure should show computed heldout score (averaged over 5 folds) as a function of the polynomial feature order.

Your plot should show two lines:

- one line for N=20 (the first 20 data points in training set)
- one line for N=512 (full training set)

Use the starter code: most of this plot is already provided.

Problem 3: Model Selection via Evidence on the Training Set

In problem 2, we needed to perform K separate fits of a linear regression model at each possible hyperparameter configuration (α, β). We'll now explore a strategy that requires only one fit of the model for each: model selection via the evidence (marginal likelihood).

Tasks for Code Implementation

Step (i) will be evaluated by autograder

CODE 3(i) Implement the `fit_and_calc_log_evidence` method in the starter code LinearRegressionPosteriorPredictiveEstimator.py. Use the formula given in Bishop PRML Eq. 3.86.

CODE 3(ii) Implement the `main` method of `run_grid_search__evidence.py`, filling in required TODOs.

Tasks for Report PDF

3a: FIGURE: Probabilistic predictions of evidence-selected models

Show a figure that visualizes probabilistic predictions on N=20 training data for your PPE models that were selected to maximize train-set evidence. Each panel shows the best model for a different polynomial feature transform {order 1, order 3, order 9}, using the alpha/beta values *selected specifically for that order*.

This will be similar to 2a above, but uses PPE instead of MAP.

Use the starter code: most of this plot is already provided.

3b: FIGURE: Model Evidence vs. Polynomial Order

Your figure should show computed per-example log evidence as a function of the polynomial feature order.

This will be similar to 2b above, but uses PPE instead of MAP.

Use the starter code: most of this plot is already provided.

Tasks for Self Reflection

You don't need to do anything for the report, but for your own understanding/learning, we encourage you to reflect on the following:

- What are some advantages of the selection strategy used in Problem 2?
- What are some advantages of the selection strategy used in Problem 3?
- How does the figure you made in 3b compare with Bishop PRML textbook's figure 3.14?

Debugging Tips

Hints for numerical stability

When you need to compute $A^{-1}x$ for some invertible matrix A and vector x, the best way to do that within numpy is to do:

```
np.linalg.solve(A, x)
```

This computes the desired product in one step, rather than first computing the inverse and then later the product. Doing the "solve" directly avoids potential inaccuracies (though for the small scale problems in this CP this is unlikely to matter too much).
