E-181 : Practical 2

Due on Friday, February 28, 2014

Kaggle Team: benni

Praphruetpong Athiwaratkun

Warm Up

In this section, we attempt to estimate the g-force on helmets with linear regression with basis functions by the maximum-likelihood method, as well as the Bayesian method.
The sets of basis functions that I use in this warm up exercise is the following

- Polynomials. Out of curiosity, I experimented with the canonical basis and Legendre polynomials.
- Sinusoidals. The wavelength in the Fourier series expansion is chosen to be twice of the maximum length in x.

Maximum Likelihood Model

Model the g-force as given by a deterministic function with additive Gaussian noise.

$$t = w^T \phi(x) + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, \beta^{-1})$ and ϕ is a basis function. The maximum likelihood parameters w is given by

$$w_{ML} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{t}$$

The variance estimated from the training set is

$$\frac{1}{\beta_{ML}} = \frac{1}{N} \sum_{n=1}^{N} \{t_n - w_{ML}^T \phi(x_n)\}^2$$

Baysian Model

In this model we treat $\vec{w}$ as a random variable with a Gaussian prior of constant variance α^{-1}. A distribution of t derived from the posterior distribution of $\vec{w}$ is Gaussian with mean and variance

$$m(x) = \beta_{ML} \phi(x)^T S \sum_{n=1}^{N} \phi(x_n) t_n$$

$$s^2(x) = \beta_{ML}^{-1} + \phi^T S \phi(x)$$

and

$$S = \alpha I + \beta_{ML} \sum_{n=1}^{N} \phi(x_n) \phi(x)^T$$

Below are the plots of data and predictions using the sinusoidal basis functions with the 7 harmonics (15 terms). The error bar indicates a distance of 1 standard deviation. The prior distribution of $\vec{w}$ is assumed to have a precision $\alpha = 0.0001$. Note that $\beta_{ML} = 0.00128$. The harmonic frequency is chosen to be $f_0 = \frac{2\pi}{2max(x)}$. That is, the wavelength of the basis function is chosen to be twice the span in x. This is to prevent the constraint that the function will need to loop back to the same value, which would be the case if λ were chosen to be $\max(x)$.

Warm Up continued on next page...

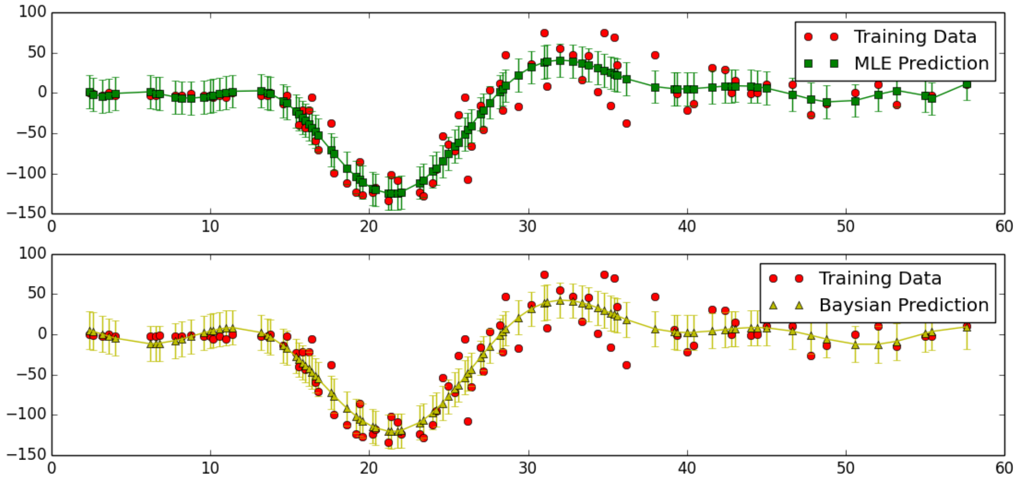

Figure 1: Training Data and Predictions for Sinusoidal Basis Functions (15 terms)

Below are the plots of the training data and the predictions using 9 terms of Legendre's polynomials. For the Bayesian's model, the prior's precision is chosen to be 0.0001 as well.

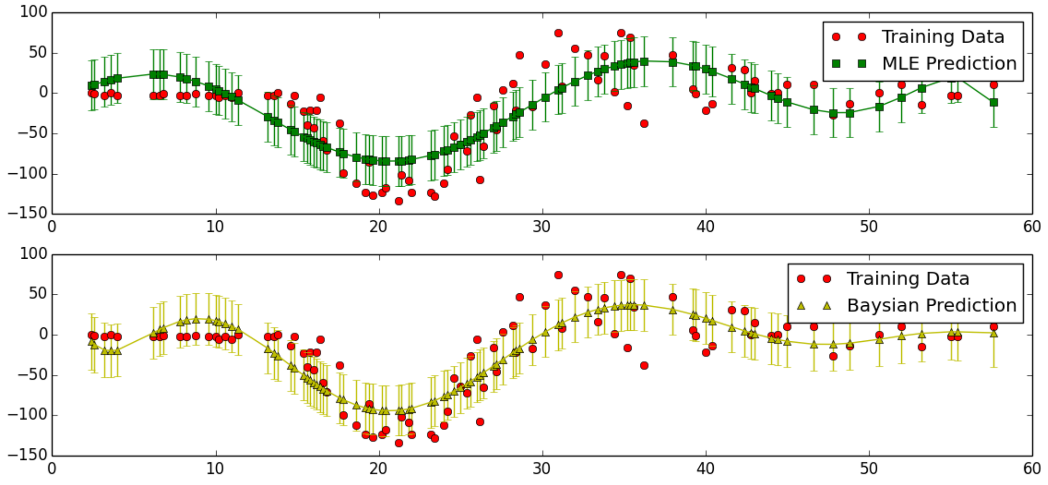

Figure 2: Training Data and Predictions for Legendre Polynomials as Basis Functions (9 terms)

Movie Revenue Prediction

Overview

Below is a high level summary of how I approach the movie revenue prediction problem.

- After observing the features that could be extracted in `trail.xml`, I decide that a neural network should be the best tool for fitting a non-linear function over a high-dimensional domain. The neural network implemented has the mean square error as the objective function. There is one hidden layer of variable size, as well as a regularization parameter for both the input and the hidden layer units.
- I extract the first set of parameters from `train.xml` which comprises of 11 features. Most of these features are in the form that are already numeric such as budget, duration, etc.
- Additional features are extracted from `train.xml` which includes genre indicator variables, production film indicator variables, etc. The total number of features is 32.
- Observing that the model with 32 features do not perform better than the preliminary model, I suspect that multicollinearity might be the culprit. I revert back to a more parsimonious set of features and carefully construct various sets of features by making sure that the additional features are meaningful. This comprises of several additional sets.

Neural Network

Rationale

For this practical, I decide to use a neural network due to its flexibility. As mentioned in Bishop (2006), a neural network can uniformly approximate any continuous function to arbitrary accuracy given sufficient hidden units. If there is an underlying pattern in the movie revenue based on the inputs provided, a neural network with high number of hidden units should be able to fit it without too much constraint, compared to other regression methods such as a linear regression with basis functions. I also use regularization to alleviate the high-variance problem. The regularization is not performed on the bias term since it could arbitrarily favors one parameter solution over an equivalent one. (Bishop, 2006) For simplicity, however, the same regularization parameter is used for both the input layer and the hidden layer units. This corresponds to the gaussian prior over parameters in which both parameter sets have the same variance.

Model

The objective function used in this model is the average square error, given by

$$E(\Theta) = \frac{1}{N}\sum_{n}^{N} \|\hat{y}(\vec{x}_n, \Theta) - y_n\|_2^2$$

where $\hat{y}(\vec{x_n}, \Theta)$ is a prediction based on the parameters Θ, $\vec{x}_n$ is an input feature vector, and y_n is revenue observed for observation n. For each input feature $\vec{x}$, the prediction given by the forward propagation is:

$$\begin{aligned}
z^{(1)} &= [1; \vec{x}] && \text{(Add 1 to the column vector } \vec{x}\text{)} \\
a^{(2)} &= (\Theta^{(1)})z^{(1)} \\
z^{(2)} &= g(a^{(2)}) \\
a^{(3)} &= \Theta^{(2)}[1; z^{(2)}] && \text{(Add 1 to the column vector } z^{(2)}\text{)} \\
\hat{y}(\vec{x_n}, \Theta) &= a^{(3)}
\end{aligned}$$

Movie Revenue Prediction continued on next page...

where $\Theta^{(1)}$ is a parameter matrix of the input layer, $\Theta^{(2)}$ is a parameter matrix of the hidden layer, and g is the sigmoid activation function given by

$$g(x) = \frac{1}{1+e^{-x}}.$$

The parameters $\Theta^{(1)}, \Theta^{(2)}$ are trained by the error backpropagation method. The output layer error $\delta^{(2)}$ and hidden layer error $\delta^{(1)}$ are given by

$$\begin{aligned}
\delta^{(2)} &= \hat{y}(\vec{x}, \Theta) - y \\
\Theta_r^{(2)} &= \Theta^{(2)}(: \ , \ 2 : \mathit{end}) && \text{(Remove the bias column)} \\
\vec{\delta}^{(1)} &= g'\left(a^{(2)}\right) .* (\Theta_r^{(2)})^T \delta^{(2)} && \text{(The product .* here means element-wise)}
\end{aligned}$$

where $\vec{y}$ is the target vector. Note that $\delta^{(2)}$ The gradients of $\Theta^{(1)}$ and $\Theta^{(2)}$ are given by

$$\vec{\nabla}_{\Theta^{(\ell)}} E = \frac{2}{N} \sum_{n=1}^{N} \delta^{(\ell)} (a^{(\ell)})^T$$

With the regularization parameter λ on both $\Theta^{(1)}$ and $\Theta^{(2)}$, the objective function becomes

$$E(\Theta) = \frac{1}{N} \sum_{n}^{N} \|\hat{y}(\vec{x}_n, \Theta) - y_n\|_2^2 + \frac{\lambda}{2N} \left\|\Theta_r^{(1)}\right\|_2^2 + \frac{\lambda}{2N} \left\|\Theta_r^{(1)}\right\|_2^2$$

where $\Theta_r^{(\ell)}$ are the reduced parameter matrices (without the bias column). The gradients of the non-bias parameters are given by

$$\vec{\nabla}_{\Theta_r^{(\ell)}} E = \frac{2}{N} \sum_{n=1}^{N} \delta^{(\ell)} (a^{(\ell)})^T + \frac{\lambda}{N} \Theta_r^{(1)} + \frac{\lambda}{N} \Theta_r^{(2)}.$$

Note that the gradients of the bias parameters are zero. Practically, the gradients computed are:

$$\vec{\nabla}_{\Theta^{(\ell)}} E = \frac{2}{N} \sum_{n=1}^{N} \delta^{(\ell)} (a^{(\ell)})^T + \frac{\lambda}{N} [\vec{0}; \Theta_r^{(1)}] + \frac{\lambda}{N} [\vec{0}; \Theta_r^{(2)}].$$

Implementation

The neural network model is done in Matlab/Octave. The regularized gradients are validated with the finite differences method. For parameter matrices $\Theta^{(l)}$ of small size, with initial values randomized to be within -1 and 1 and a finite step of $\epsilon = 10^{-4}$, the sum of square differences is $\approx 10^{-11}$. The default number of iterations for gradient descent is set to be $10,000$. This is sufficiently fast for small number of features. (less than a minute for input size of 11 and hidden layer size of 9, for instance)

Experiments

This section explains the feature extraction and the result for some of the experiments. I choose a few representative examples to explain in details.

Set 1

Feature Extraction

Below is the list of features.

1. Duration in minutes
2. Number of theatres
3. Rating from 0 to 4 (before scaling). The mapping is $\{G \to 0,\ PG \to 1,\ PG-13 \to 2,\ R \to 3,\ NC-17 \to 4\}$. I decide to map all the ratings to one variable because it is naturally ranked from most general G to most restrictive *NC-17*. The neural network should also be able to fit a non-linear pattern if that is the case.
4. Budget
5. The number of Oscar winning directors
6. The number of highest grossing actors
7. The number of Oscar winning actors
8. Summer release indicator
9. Christmas release indicator
10. Memorial day release indicator
11. Labor day release indicator

Note: As in other feature sets, all extracted values are floating numbers, scaled to have sample mean of 0.0 and sample variance of 1.0 for each feature.

Results

I did a neural network training using several hidden layer sizes. I tend to prefer higher hidden layer size as it offers more flexibility. The optimal regularization term seems to be naturally heavier than in smaller hidden layer model. Below is a plot of a mean absolute error (MAE) versus $\log(\lambda)$ for 30 hidden units.

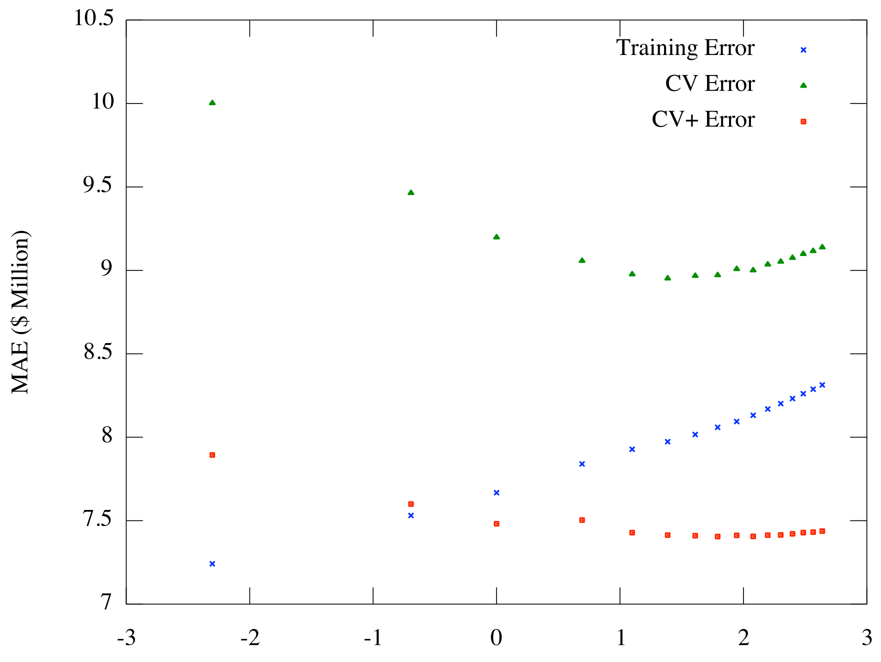

Figure 3: Mean Absolute Error Versus $\log(\lambda)$.

The training error computed is the mean absolute error of the training sample, which is a randomized 66% of the original data set. The CV+ error is the cross validation error after converting negative revenues to zero as we know that actual revenues are bounded below. Below is an example of prediction based on a model with hidden layer size 9 and $\lambda = 7$.

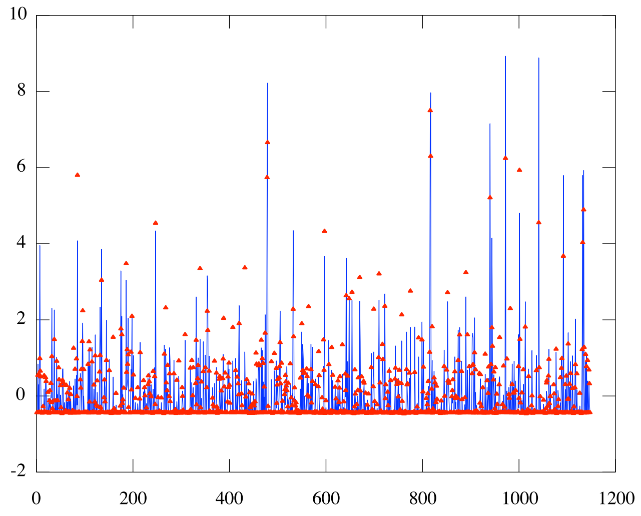

Figure 4: Log of Actual Revenues and Predicted Revenues

To find the optimal hidden layer size and the regularization parameter λ, I run several batches of neural network training and produce plots as in Figure 3. For each batch, the observations are randomly permuted and split into the training set and cross validation set.

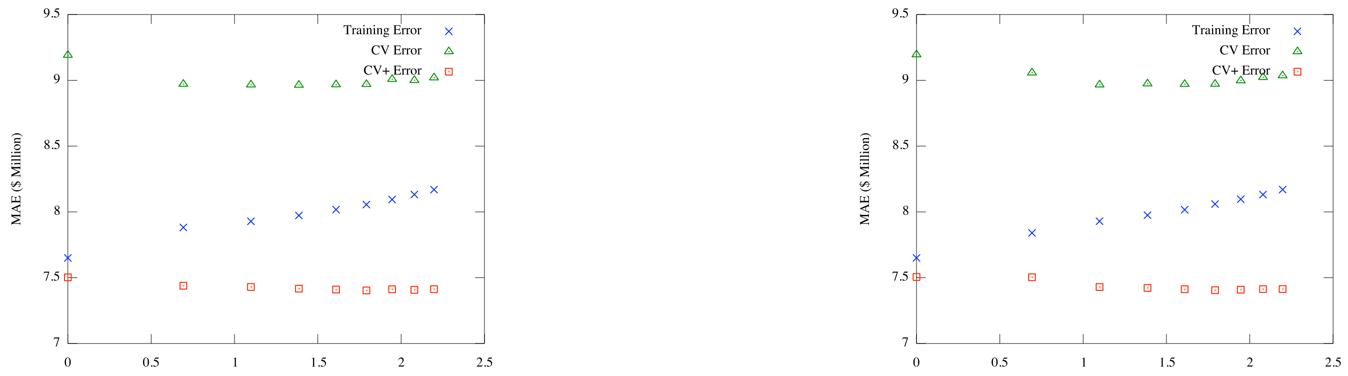

Figure 5: Training and Cross Validation Sets for Set 1. The hidden layer size = 30.

I submit a few predictions based on this set. For hidden layer size of 30, it seems optimal to use $\lambda = 6$. This gives me a public score of $3,630,410$ on Kaggle Leaderboard.

Discussion

The cross validation error seems to depend heavily on the regularization parameter. It also heavily depends on the data in the training set as I observe wildly different results across random permutations.

Set 2

I aim to improve the prediction by including other explanatory variables. This set extends the features of **Set 1** to include production company indicators, best seller indicator, month, the genre indicators.

Feature Extraction

The total number of features in this set is 32. They contain **Set 1** and the following:

- Month (1 to 12)
- Indicator of whether the string 'best-sell' exists in the review
- Major production company indicators. One indicator for each of the following production companies: Miramax, DreamWorks, Paramount, Twentieth Century Fox, Universal, Sony, Warner Brothers.
- 12 genre indicators.

After the features are extracted, they are scaled to be have sample mean 0 and sample variance 1.

Results

Similar to set 1, I train the parameters using several hidden layer size. On average, the training error and the cross validation error seems noticeably high for this set compared to Set 1.

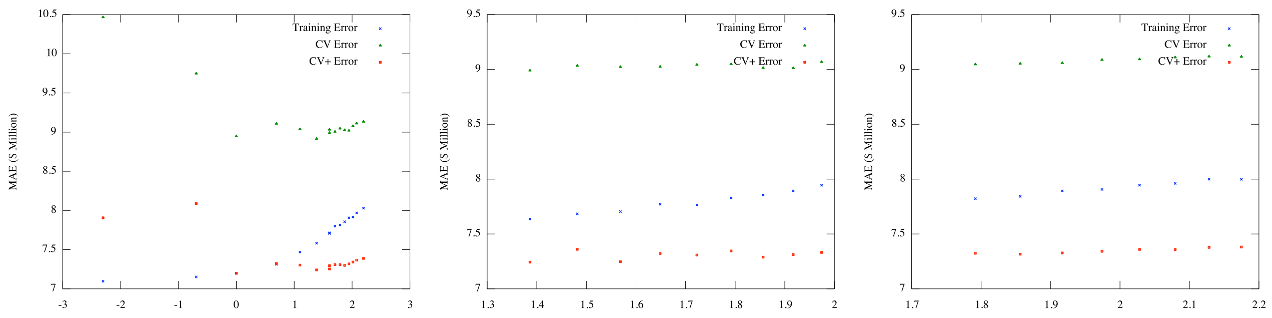

Figure 6: Mean Absolute Error of Training and Cross Validation Sets Versus log(λ)

The cross validation error for this set seems to be higher than in **Set 1**. A submission to Kaggle for a model with hidden layer size of 30 and $\lambda = 6$ earns a public score of $3,882,779$, which is higher than the best score for set 1. While I realize that the public score might be deceiving, I have an impression that choosing input features can be important as features that are not meaningful could adversely affect the robustness of the fit. This movivate me to experiment with feature extractions in later models.

Additional Sets

Set 3

Features in this set consists of those in **Set 1** and the following:

12. Month (1 to 12)

22. Indicator of whether the string 'best-sell' exists in the review

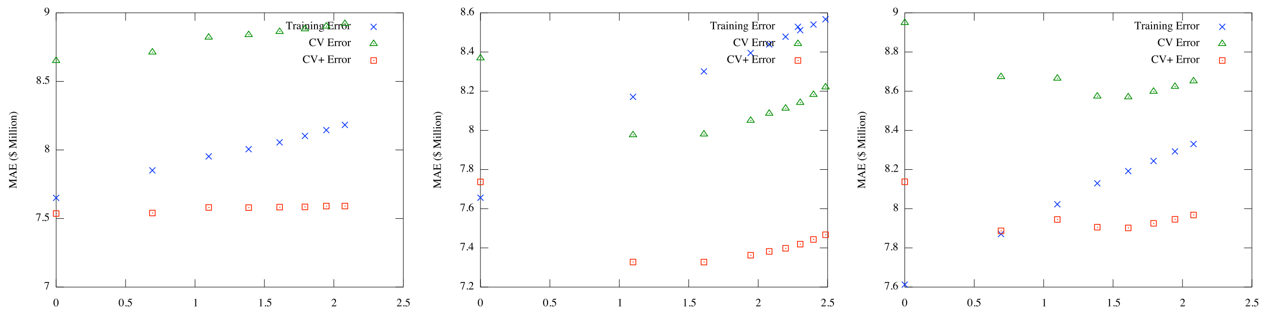

Figure 7: Training and cross validation errors for **Set 3**. Hidden input size = 30.

Set 4

The features in **Set 4** consists of features in **Set 3** and the following:

14. One indicator of whether the production film is one of the seven companies (Miramax, DreamWorks, Paramount, Twentieth Century Fox, Universal, Sony, Warner Brothers.)

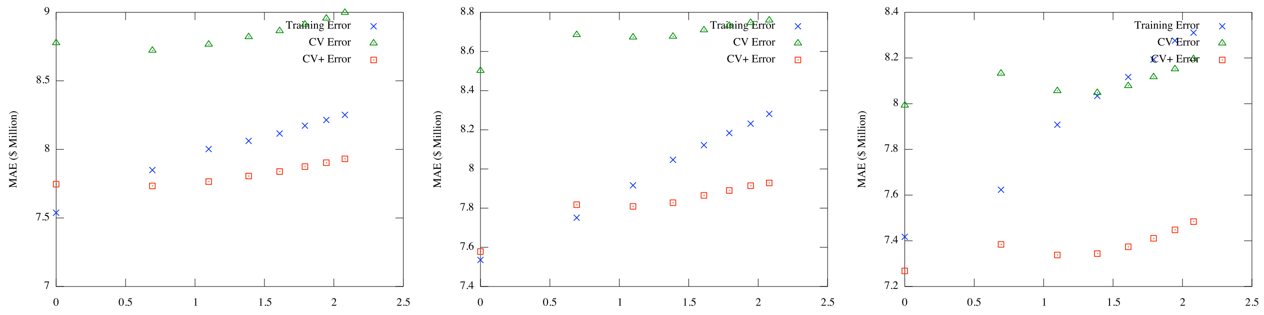

Figure 8: Training and cross validation errors for **Set 4**. Hidden input size = 30.

Set 5

The features in **Set 5** are the ones in **Set 4** and a small subset of genres that seems to be most representative.

15. Horror or suspense genre indicator. (Set to 1 if one of its genres is horror or suspense, otherwise 0)

25. Fantasy or adventure genre indicator (Set to 1 if one of its genres is fantasy or documentary, otherwise 0)

35. Documentary genre indicator

45. Drama genre indicator

55. Romance genre indicator

65. Comedy genre indicator

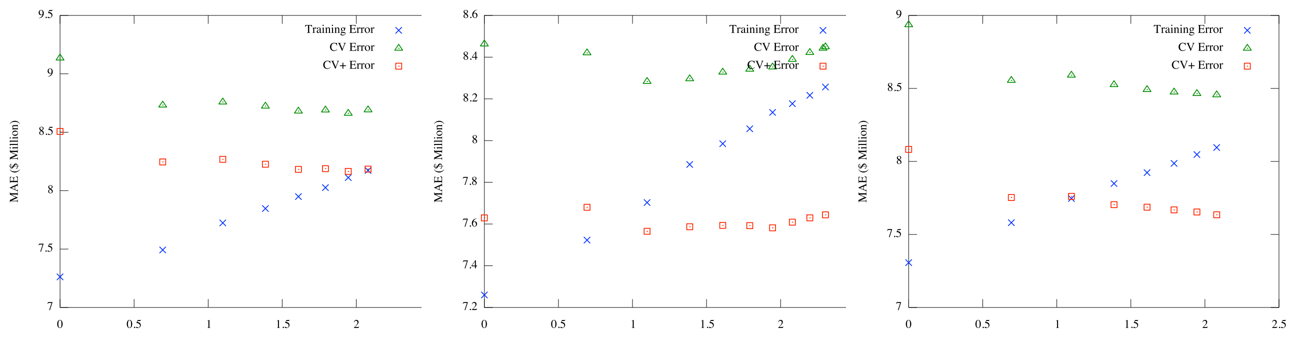

Figure 9: Training and cross validation errors for **Set 5**. Hidden input size = 30.

Summary

For each set, I attempt to choose an optimal regularization parameter and a hidden layer size. The hidden layer size is quite computationally expensive to loop through since the optimal regularization needs to be found for each size as well. Therefore, I often fix the hidden layer size and iterate to find the optimal λ for a given size. Below summarizes the Leaderboard scores for each set.

Table 1: Mean Absolute Errors for Various Feature Sets

Set	Number of Input Features	Hidden Layer Size	λ	Leaderboard Score
1	11	9	7	3,675,021
2	32	32	6	3,840,340
3	13	30	5	3,522,931
4	14	30	5	3,486,408
5	20	30	5	3,695,387

In this model I use the mean square error as the objective function. If the data do not have extreme outliers, the mean square error should be roughly equivalent to the mean absolute error as an cost function. However, the movies with outstandingly high revenues can cause the parameters to unneccessarily skew towards predicting higher revenue for the movies with similar features in the case of mean square error cost function.