Imperial College
London

EE3-23 Machine Learning

IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

Assignment 3

Anand Kasture (CID: 00832896)

Date: March 10, 2017

Pledge

1 Approximate ERM

In this section, we will derive an upper bound on the difference between the out-of-sample error associated with the ϵ-optimal hypothesis and some test error minimising hypothesis chosen from the hypothesis set $\mathcal{H}$ i.e. $R(h_\epsilon) - R(h^*)$

We will first invoke the definition of the ϵ-optimal hypothesis as presented in the question:

$$\hat{R}_n(h_\epsilon) \leq \epsilon + \hat{R}_n(g) \tag{1}$$

Hereafter, we invoke the Vapnik-Chervonenkis bound as seen in the lectures. The following inequality holds with probability at least $1 - \delta_1 - \delta_2$:

$$R(g) \leq R(h^*) + \sqrt{\frac{8d_{VC}(\mathcal{H})}{n}\log(2n+1) + \frac{8}{n}\log\frac{4}{\delta_1}} + \sqrt{\frac{1}{2n}\log\frac{2}{\delta_2}} \tag{2}$$

Notice the presence of the $\hat{R}_n(g)$ and $R(g)$ terms in Equation (1) and Equation (2) respectively. In order to obtain an upper bound on $\epsilon + \hat{R}_n(g)$, we must first acquire a lower bound on $R(g)$. We will use the one-sided variant of Hoeffding's inequality to achieve this. Equation 3 shows the resulting expression.

$$\hat{R}_n(g) - \sqrt{\frac{1}{2n}\log\frac{1}{\delta}} \leq R(g) \tag{3}$$

Carrying out (3) $\rightarrow$ (2), we achieve:

$$\hat{R}_n(g) - \sqrt{\frac{1}{2n}\log\frac{1}{\delta}} \leq R(g) \leq R(h^*) + \sqrt{\frac{8d_{VC}(\mathcal{H})}{n}\log(2n+1) + \frac{8}{n}\log\frac{4}{\delta_1}} + \sqrt{\frac{1}{2n}\log\frac{2}{\delta_2}}$$

$$\hat{R}_n(g) \leq R(h^*) + \sqrt{\frac{8d_{VC}(\mathcal{H})}{n}\log(2n+1) + \frac{8}{n}\log\frac{4}{\delta_1}} + \sqrt{\frac{1}{2n}\log\frac{2}{\delta_2}} + \sqrt{\frac{1}{2n}\log\frac{1}{\delta}} \tag{4}$$

Next, (4) $\rightarrow$ (1), produces the following expression:

$$\hat{R}_n(h_\epsilon) \leq \epsilon + \hat{R}_n(g) \leq \epsilon + R(h^*) + \sqrt{\frac{8d_{VC}(\mathcal{H})}{n}\log(2n+1) + \frac{8}{n}\log\frac{4}{\delta_1}} + \sqrt{\frac{1}{2n}\log\frac{2}{\delta_2}} + \sqrt{\frac{1}{2n}\log\frac{1}{\delta}} \tag{5}$$

Equation (5) is written in $\hat{R}_n(h_\epsilon)$. We will now obtain an expression involving $R(h_\epsilon)$ through the use of Hoeffding's inequality once again. The resulting expression is shown in Equation (6)

$$R(h_\epsilon) - \sqrt{\frac{1}{2n}\log\frac{1}{\delta}} \leq \hat{R}_n(h_\epsilon) \tag{6}$$

Finally, carrying out (6) $\rightarrow$ (5), and rearranging gives the upper bound we set out to achieve.

$$R(h_\epsilon) - \sqrt{\frac{1}{2n}\log\frac{1}{\delta}} \leq \hat{R}_n(h_\epsilon) \leq \epsilon + R(h^*) + \sqrt{\frac{8d_{VC}(\mathcal{H})}{n}\log(2n+1) + \frac{8}{n}\log\frac{4}{\delta_1}} + \sqrt{\frac{1}{2n}\log\frac{2}{\delta_2}} + \sqrt{\frac{1}{2n}\log\frac{1}{\delta}}$$

$$R(h_\epsilon) - R(h^*) \leq \epsilon + \sqrt{\frac{8d_{VC}(\mathcal{H})}{n}\log(2n+1) + \frac{8}{n}\log\frac{4}{\delta_1}} + \sqrt{\frac{1}{2n}\log\frac{2}{\delta_2}} + 2\cdot\sqrt{\frac{1}{2n}\log\frac{1}{\delta}}$$

2 Optimal Predictor for the Absolute Error

3 Movie recommendation

Part A - Constant Base Predictors

Table 1 illustrates the training and test performance of two predictors that generate fixed ratings based on mean training data ratings. The two predictors produce a constant rating for each user (movie-independent), and a constant rating for each movie (user-independent) respectively. These results can be replicated by running the `constBasePredictors.m` script.

	Training Error	**Test Error**
Constant User Rating	0.910818	**0.926870**
Constant Movie Rating	**0.785146**	0.979210

Table 1: Training, Test Error Performance for Constant Ratings Predictors

The constant movie ratings predictor records a comparatively better training error. In the real world, we only have access to a training data set and can only make decisions based on the in-sample error. This leads us to selecting the constant movie ratings predictor. However, this predictor demonstrates poor generalisation to test data, with an out-of-sample error that is higher than that for the constant user ratings predictor. Therefore, the constant user ratings predictor is the preferred choice in hindsight.

Part B - Linear Regression Baseline

We will now estimate the user ratings using linear regression in conjunction with techniques known to tackle over-fitting. Consider the following optimisation problem: $\min \frac{1}{N}\sum_{n=1}^{N}(r_n - \mathbf{w}^T\mathbf{v}_n)^2$. We would like to find the best weight vector $\mathbf{w}^\star_{lin}$ for *each* user such that the mean squared training error is minimised. Here, N represents the number of movies reviewed by the current user in the training data set, r_n represents a single rating, and $\mathbf{v}_n$ represents the corresponding feature vector.

Using linear regression, we can obtain a closed form solution to the aforementioned problem as follows: $\mathbf{w}^\star_{lin} = \left(\mathbf{V}^T\mathbf{V}\right)^{-1}\mathbf{V}^T\mathbf{r}$ where $\mathbf{V} = [\mathbf{v}_1 \;\; \mathbf{v}_2 \;\; \dots \;\; \mathbf{v}_N]^T$ and $\mathbf{r} = [r_1 \;\; r_2 \;\; \dots \;\; r_N]^T$. Regularisation refers to a technique where we add a constraint on the weights in order restrict the range of values they can take up. This approach is known to help counter the potential over-fitting of training data and is also motivated by the occurrence of ill-conditioned feature matrices $\mathbf{V}$.

Adding an upper bound on $\mathbf{w}^T\mathbf{w}$ yields the following optimisation problem: $\min \frac{1}{N}\sum_{n=1}^{N}(r_n - \mathbf{w}^T\mathbf{v}_n)^2 + \lambda\mathbf{w}^T\mathbf{w}$. The linear regression solution is now given by $\mathbf{w}^\star_{reg} = \left(\mathbf{V}^T\mathbf{V} + \lambda\mathbf{I}\right)^{-1}\mathbf{V}^T\mathbf{r}$. Note that the choice of the exact regularisation function is largely heuristic. On the contrary, the choice of λ is extremely principled, and is chosen through cross validation.

In machine learning, validation refers to an optimal model selection technique where the training data set $\mathcal{D}$ is partitioned into two distinct segments: $\mathcal{D}^{train}$ and $\mathcal{D}^{val}$. However, there exists a trade-off between obtaining an accurate hypothesis estimate (large $\mathcal{D}^{train}$) and obtaining an accurate out-of-sample error estimate (large $\mathcal{D}^{val}$). The K-fold cross validation technique addresses this by segmenting the training data into K blocks. The best performing model produces the smallest mean square validation error across all $\{\mathcal{D}^{train}, \mathcal{D}^{val}\}$ configurations. In each one of the K configurations, $\mathcal{D}^{val}$ is set to a single (different) block, and $\mathcal{D}^{train}$ is set to the remaining K-1 blocks. The cross validation error is considered a good indicator of how well the model will fit out-of-sample data.

We will investigate two different 10-fold cross validation implementations in selecting the λ parameter. The `linspace` function was used to select $\lambda^\star$ from 100 different values in each case.

- Constant $\lambda^\star \in [0.01, 2]$ for every $\mathbf{w}^*_{reg}$ calculation (`getLambdaAllUsers.m`)
- Variable $\lambda^\star \in [0.01, 5]$ for every $\mathbf{w}^*_{reg}$ calculation (`getLambdaPerUser.m`)

Table 2 shows the training and test performance of the ridge regression approach when a constant $\lambda^\star$ is used to construct the feature vector for every unique user in the training set. Results for two different movie feature vector normalisation techniques (`normc` and `zscore`) are also shown. Figure 1 illustrates the cross validation error and training error plots that are generated when selecting $\lambda^\star$.

	nominal	**normc**	**zscore**
Training Error	0.742776	0.755268	**0.712409**
Test Error	0.936477	**0.896776**	1.077042
$\lambda^\star$	1.3769	1.4372	0.3919

Table 2: Training, Test Error Performance for Ridge Regression with constant $\lambda^\star$

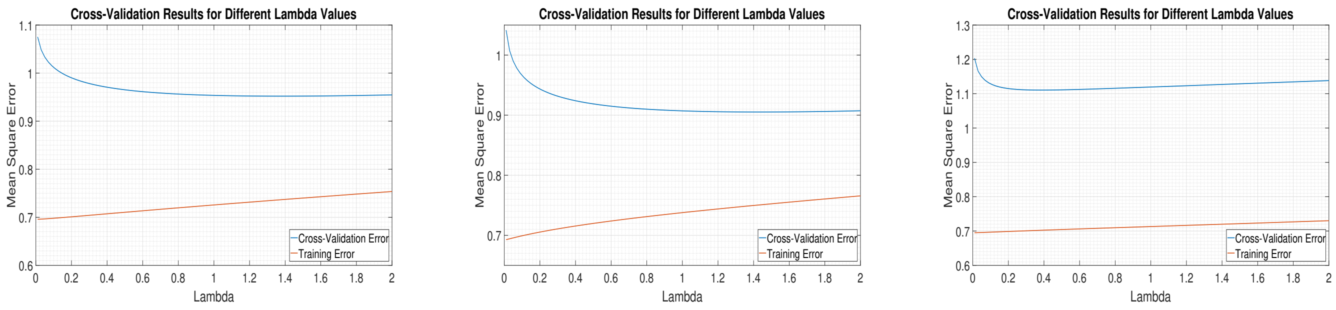

Figure 1: Cross-Validation Error Plots for nominal, normc and zscore

Table 3 shows same cross-validation results when a variable $\lambda^\star$ is used. Figure 2 illustrates the (Rayleigh) probability distribution fits on the $\lambda^\star$ values for each user.

	nominal	**normc**	**zscore**
Training Error	0.742756	0.751678	**0.719734**
Test Error	0.946441	**0.905369**	1.066379

Table 3: Training, Test Error Performance for Ridge Regression with variable $\lambda^\star$

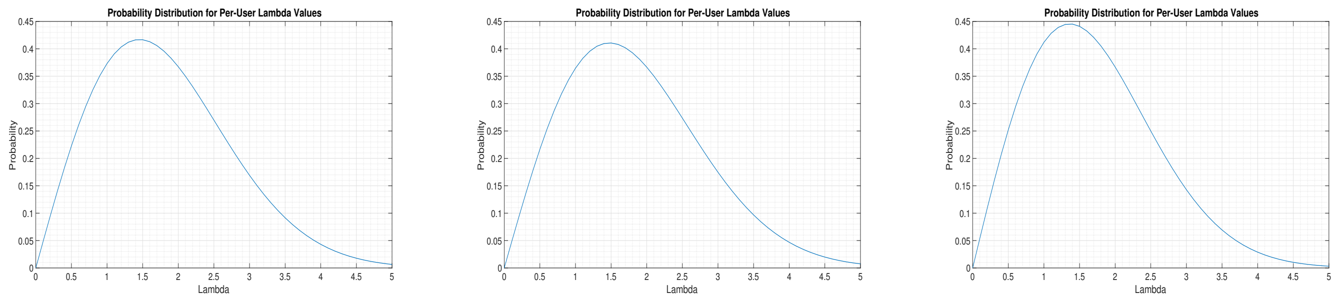

Figure 2: Probability Distribution for variable $\lambda^\star$ for nominal, normc and zscore

In general, the variable $\lambda^\star$ strategy does not make a big impact on the error results. This can be explained by noting the heightened unreliability of the cross-validation technique for users with a relatively small number of movie ratings. The best test performance was recorded with the constant $\lambda^\star$ strategy using `normc` normalised feature vectors. These results show that one can achieve a definitive (albeit limited) improvement upon the constant base predictors by attempting to best fit the training data.

Part C - Transformed Features

In this section, we will evaluate the effect a standard non-linear transformation of the feature set has on training and test error performance. The transform shown below uses a quadratic expansion to map the features to a sequence of monomials. This is implemented using the `x2fx(V,'quadratic')` function from the Statistics and Machine Learning Toolbox.

$$\begin{bmatrix} x_1 & \dots & x_{18} \end{bmatrix} \xrightarrow{\Phi_Z(\cdot)} \begin{bmatrix} 1 & x_1 & \dots & x_{18} & x_1x_2 & \dots & x_{17}x_{18} & x_1^2 & \dots & x_{18}^2 \end{bmatrix}$$

Table 4 repeats the set-up used in producing Table 2, except now we are now using the higher dimensional transformed movie features within the computation.

	nominal	**normc**	**zscore**
Training Error	0.556823	0.666486	**0.467228**
Test Error	1.038890	**0.911428**	1.589808
$\lambda^\star$	1.3568	1.4372	0.3517

Table 4: Training, Test Error Performance for Ridge Regression with transformed movie features

The results in 4 illustrate worse in-sample to out-of-sample generalisation in comparison to the non-transformed case. Namely, there is a noticeable decrease in the training error and an increase in the test error across all 3 normalisation methods. This can be explained as follows: The non-linear transform $\Phi_Z : \mathbb{R}^{18} \to \mathbb{R}^{190}$ significantly increases the dimensionality of the feature space. This allows for a tighter training data fit using ridge regression. However, despite the use of a cross-validated $\lambda^\star$, the results point to an increased over-fitting effect as a consequence of the transform.

Note that the original feature data set is a large and sparse binary matrix. Consequently, a large proportion of the monomials in the transformed feature vector are driven to the value 0. Therefore, the effective VC-Dimension of this hypothesis set is likely to be lower than expected with the above set up, and an alternative approach that takes the matrix structure into account is likely to improve performance.

Part D - Collaborative Filtering

In this section, we are tasked with constructing user and movie features using collaborative filtering. This technique is built upon the notion that users with similar tastes (inferred from the existing training data ratings) are likely to rate unseen movies in a similar manner. We invoke the stochastic gradient descent (SGD) method to solve this problem. In SGD, we apply the gradient descent algorithm to a randomly selected training sample. This modification is justified since the average search direction equates to the gradient of the training error i.e. the quantity we are trying to minimise. We formulate the movie ratings problem as follows

Let $\mathbf{u}_i, \mathbf{v}_j \in \mathbb{R}^K$ represent the weights for user $i \in [1, 671]$ and movie $j \in [1, 9066]$, and r_{ij} the corresponding training data rating.

$$\min_{\mathbf{u}_i, \mathbf{v}_j} \left(r_{ij} - \mathbf{u}_i^T \mathbf{v}_j \right)^2 + \lambda \left(\|\mathbf{u}_i\|^2 + \left\|\mathbf{v}_j\right\|^2 \right)$$

We will now derive the SGD equations for both the user and movie features. Note that $\mathcal{L}$ represents the cost function, and γ denotes the learning rate. The features are initialsed to random values in [0,1].

$$\mathbf{u}_i^{t+1} = \mathbf{u}_i^t - \gamma \frac{\partial \mathcal{L}\left(r_{ij}, \mathbf{u}_i^t, \mathbf{v}_j^t\right)}{\partial \mathbf{u}_i^t} \quad , \quad \mathbf{v}_i^{t+1} = \mathbf{v}_i^t - \gamma \frac{\partial \mathcal{L}\left(r_{ij}, \mathbf{u}_i^t, \mathbf{v}_j^t\right)}{\partial \mathbf{v}_i^t}$$

$$\mathbf{u}_i^{t+1} = \mathbf{u}_i^t + \gamma\left(\left(r_{ij} - \left(\mathbf{u}_i^t\right)^T \mathbf{v}_j^t\right)\mathbf{v}_j^t - \lambda\mathbf{u}_i^t\right)$$

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \gamma\left(\left(r_{ij} - \left(\mathbf{u}_i^t\right)^T \mathbf{v}_j^t\right)\mathbf{u}_j^t - \lambda\mathbf{v}_i^t\right)$$

Selecting γ

The precise value of the SGD step length is often a heuristic choice in line with the iteration number and/or the magnitude of the gradient. In our work, we choose to maintain a constant value of $\gamma = 0.01$ and focus on acquiring optimal $K^\star$ and $\lambda^\star$ values through cross validation instead.

Selecting K and λ

The optimal values of K and λ were chosen using cross-validation. Note that this approach scales poorly with the total number of model configurations. With time constraints in mind, we select $\lambda^\star \in \{0.0001, 0.001, 0.01, 0.1, 1\}$, and $K^\star \in \{1, 5, 10, 15\}$.

Figure 3 illustrates the cross-validation error acquired for all 20 models comprising of every K and λ combination. The error is plotted against epochs i.e. number of complete iterations through the training data set. Each coloured line plot corresponds to one value of K. It can be seen that the yellow lines associated with $K = 15$ are capable of producing the best cross-validation error results as long as iterations are stopped before the model starts over-fitting the data.

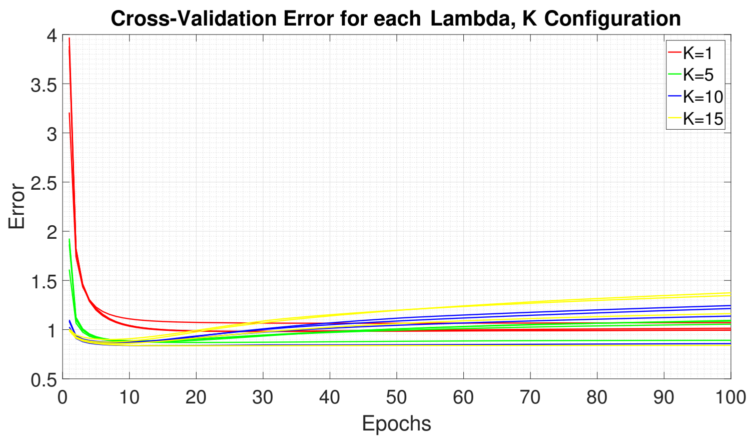

Figure 3: Cross-Validation Error for optimal model selection

Note that despite our valiant attempt at acquiring optimal model parameters, the inherent randomness in the initialisation of the feature vectors tends to affect the mean squared training and test error values. Notwithstanding this observation, we run our algorithm for the optimally choice of $K^\star = 15$ and $\lambda^\star = 0.01$. Figure 4 illustrates a plot of the training and test errors against epochs. The training error and test error plots move in roughly opposite directions after an initial drop.

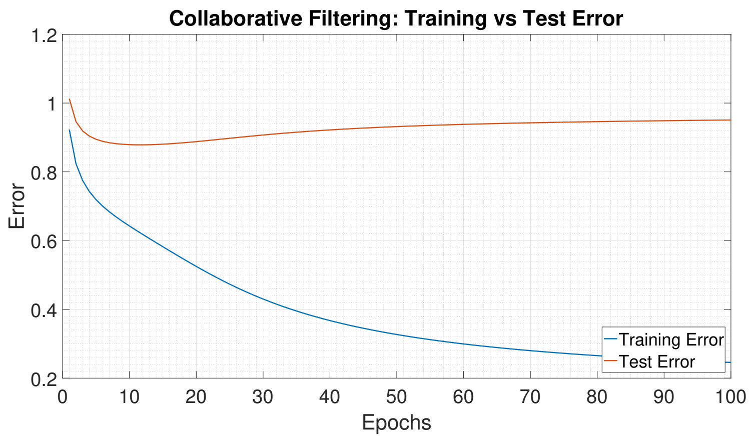

Figure 4: Collaborative Filtering: Training vs Test Error