CMPS242 Homework #1 – Polynomial Learning

Andrew Stolman

&

Zayd Hammoudeh

October 3, 2017

1 Homework Objective

The goal of this homework is to implement an algorithm that can learn the scalar weights of a 9^{th}-order polynomial function with a single independent variable. We also study the effect of a regularizing constant, λ, on over-fitting.

2 Experiment Setup

We implemented our solver in both Python (with NumPy) and Matlab; the latter implementation provides better results so only Matlab results are discussed.

2.1 Definition of $\mathbf{w}^\star$

Certain n-degree polynomials can be modeled by the simple univariate vector space, P. For the independent variable, x, the vector space's basis,

$$basis(P) = \langle 1, x, ..., x^n \rangle, \tag{1}$$

is dimension $n+1$. Given a member of the vector space, $\mathbf{x} \in P$, the resulting polynomial function is:

$$y = \mathbf{w}^{\mathrm{T}}\mathbf{x}, \tag{2}$$

where $\mathbf{w}$ is a vector of scalar weights with dimension $n+1$.

Given a set of examples, (x_i, t_i), $i = 1...m$, define $\mathbf{X}$ as an $(n+1)$ by m matrix whose i^{th} column is the vector in space P that corresponds to example value x_i. Define $\mathbf{t}$ as the vector whose value at index i is t_i. The linear system in Eq. (2.1) then becomes:

$$\mathbf{y} = \mathbf{w}^{\mathrm{T}}\mathbf{X} \tag{3}$$

As proven in the homework description, the weight vector, $\mathbf{w}^\star$, that minimizes the regularized error is

$$\mathbf{w}^\star = (\mathbf{X}\mathbf{X}^{\mathrm{T}} + \lambda I)^{-1}\mathbf{X}\mathbf{t}, \tag{4}$$

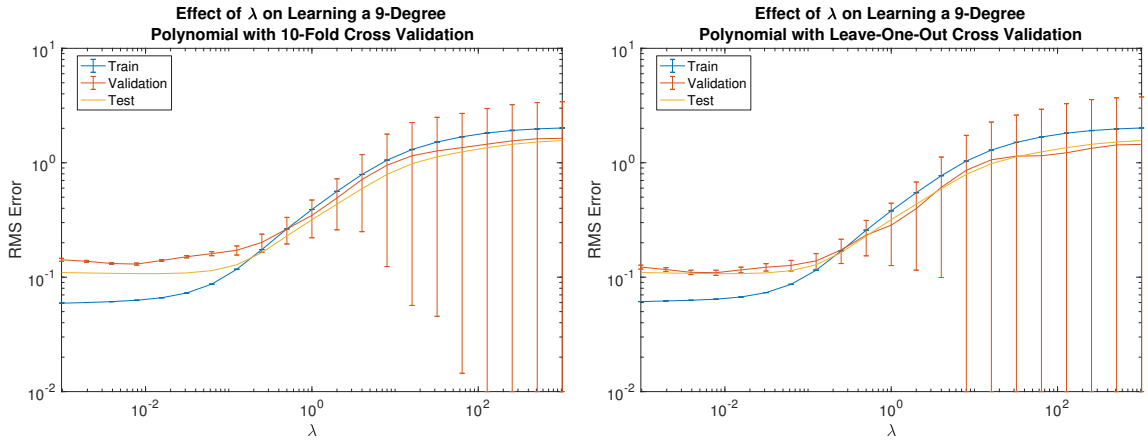
where λ is a regularizing constant and I is the $(n+1)$ identity matrix.

2.2 Improving the Calculation of $\mathbf{w}^\star$ through Linear Solving

In the homework dataset, values in $\mathbf{X}$ range approximately from $2 * 10^{-7}$ to $8 * 10^{8}$. Hence, calculating $\mathbf{w}^\star$ is greatly affected by floating point errors. As such, when trying to find the inverse matrix $(\mathbf{X}\mathbf{X}^{\mathrm{T}} + \lambda I)^{-1}$, Matlab warns that the matrix is "`near singular or badly scaled`" and further warns the "results may be inaccurate." Python reports a similar warning. As such, rather than finding $\mathbf{w}^\star$ through straight matrix multiplication, our implementation instead finds $\mathbf{w}^\star$ by solving the linear system:

$$(\mathbf{X}\mathbf{X}^{\mathrm{T}} + \lambda I)\mathbf{w}^\star = \mathbf{X}\mathbf{t} \tag{5}$$

This approach generally has lower error for this homework (by up to two orders of magnitude when $n = 19$). From a theoretical perspective, our revised technique is equivalent.



(a) $k = 10$ (b) Leave-One-Out

Figure 1: Effect of λ and Cross Validation Fold Count when Learning a 9^{th}-Order Polynomial

3 Effect of the Regularizer λ

The regularizer, λ, in Eq. (2.1) prevents overfitting by preferring models with smaller values of $\|w\|$. In our experiments, we tested different values of λ from the set, $\{0\} \cup \{2^i | i = -10...10\}$. Figure 1(a) shows the training, validation, and test errors with these values of λ for 10-fold cross-validation. For training and validation, the line represents the mean RMS-error while the bars are the variance. The x and y axes are logarithmic for clearer visualization.[1]

3.1 Selecting the Optimal Regularize

The selected value of the regularizer, λ, should minimize or nearly minimize the validation error. Table 1 shows the relationship between training, validation, and test errors for different values of λ when performing 10-fold cross validation; optimal values are **bolded**. Only values of $\lambda \leq 2^{-6}$ are included in this table as Figure 1(a) showed this range had the best overall performance.

When a regularizer is not used (i.e., $\lambda = 0$), the learning algorithm performs sub-optimally as indicated by the higher test error (0.117). The regularizer, $\lambda = 2^{-8}$, had the minimum validation error for 10-fold CV; its test error is also close to, but not exactly, the minimum.

Table 1: Learning errors for different values of λ for 10-fold cross-validation

λ	0	2^{-10}	2^{-9}	$\mathbf{2^{-8}}$	2^{-7}	2^{-6}	2^{-5}
Training	**0.057**	0.059	0.060	0.061	0.063	0.066	0.073
Validation	0.198	0.143	0.138	**0.132**	0.133	0.141	0.148
Test	0.117	0.110	0.109	0.108	**0.107**	**0.107**	0.109

[1]For this dataset, there are multiple values of λ where the validation variance exceeded its associated mean. In such cases, the standard error bar plotting technique would have indicated the occurrence negative error, which is both impossible and non-graphable when using a logarithmic scale. As such, we chose to visualize this phenomenon by clamping all negative variance bars at 10^{-2}. We found this was clearer than leaving those negative error bars out entirely.

Table 2: Learning errors for different values of λ for leave-one-out cross-validation

λ	0	2^{-10}	2^{-9}	2^{-8}	$\mathbf{2^{-7}}$	2^{-6}	2^{-5}
Training	**0.059**	0.061	0.062	0.063	0.064	0.067	0.073
Validation	0.165	0.123	0.117	0.110	**0.109**	0.116	0.122
Test	0.117	0.110	0.109	0.108	**0.107**	**0.107**	0.109

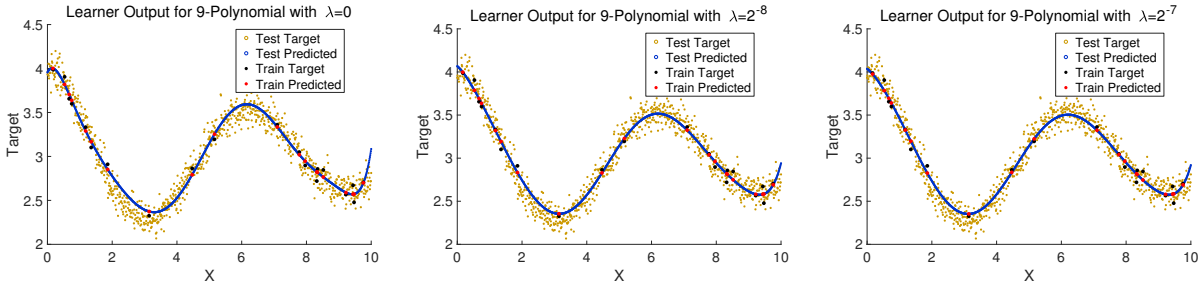
3.2 Cross-Validation Fold Count

Figures 1(a) and 1(b) shows a comparison of the 10-fold and leave-one-out (LOO) cross-validation (CV) results respectively. As expected, LOO CV has higher variance in particular at large values of λ; this can be seen visually since the error bars are longer. The reason for the increase is because when the validation set is smaller in size, the validation error of outliers becomes more prominent. Despite that, LOO CV produces better results because first the mean validation error decreased and more importantly is closer to the actual test error.

Table 2 shows a comparison of the training, validation, and test errors when performing LOO CV; optimal values are again shown in **bold**. Note that unlike 10-fold CV, LOO CV resulted in the selection of the optimal value of λ.

4 Visualizing the Learner Outputs

As mentioned previously, our system learns a polynomial function. Figure 2 compares the target (i.e., actual) and learned/predicted values for both the training and test sets. Note that subfigures (a), (b), and (c) are for λ values of 0, 2^{-8}, and 2^{-7} respectively. These result visually align with the RMS errors reported in sections 3.1 and 3.2. Recall that $\lambda = 0$ had a suboptimal test error; the cause of this can be seen in the slight hook in the graph when x is close to zero as well as the overshoot at the local maximum around $x = 6$. Similarly, the results for λ equal to 2^{-8} and 2^{-7} are very similar with no perceptible differences; this is expected as their test errors differed by less than 0.6%.



(a) $\lambda = 0$ (b) $\lambda = 2^{-8}$ (c) $\lambda = 2^{-7}$

Figure 2: Comparison of the target and predicted values for different values of λ

5 Regularizing Without Bias

We repeated the above process but modified the error function so that the constant term of the polynomial model would not be penalized. This is achieved by calculating the following error function:

$$Err(\mathbf{w}) = ||\mathbf{X}^T\mathbf{w} - \mathbf{t}||^2 + \lambda\langle\mathbf{0}_1, \mathbf{w}\rangle^2$$

where $\mathbf{0}_1$ refers to the vector with a zero in this first index and 1 everywhere else. Taking the derivative, setting it zero, and solving gives us

$$\mathbf{w}^\star = (\mathbf{X}\mathbf{X}^{\mathrm{T}} + \lambda I^*)^{-1}\mathbf{X}\mathbf{t},$$

where I^* is the identity matrix with the first row set to all zeros.

The results are summarized in Figure 3. Note that the variance and average RMS error both decreased when there was no bias regularization; the shift was particularly pronounced at larger values of λ. The ideal value of λ also shifted higher to approximately 1.

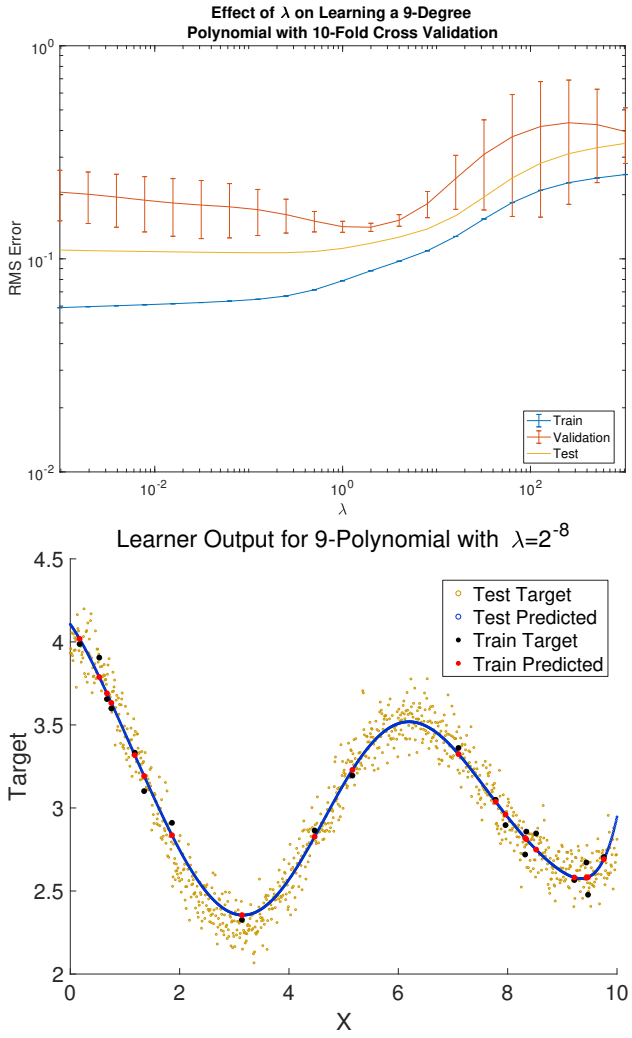


Figure 3: Results for estimation without the bias term