Gradient Based Optimization of Hyperparameters and its Application in Linear Regression

Problem Statement

Using the linear model $f(x; w) = w^T x$ the aim is to solve the regression problem and tune the hyperparameters using gradient descent.

Given the data $D = (x_i, y_i)_{i=1}^n$ for the regression problem, where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, we solve the following formulation:

$$\min \frac{1}{2}\sum_{i=1}^{n}(f(x_i; w) - y_i)^2 + \frac{1}{2}\sum_{j=1}^{d}\lambda_j w_j^2$$

where $\lambda = (\lambda_1, \lambda_2, ..., \lambda_d)^T$ are tunable non-negative hyperparameters and $w = (w_1, w_2, ..., w_d)^T$.

Problem Formulation

For training the Linear Regression model, we use the loss function Q with the following notations which will be consistent throughout this article:

$$Q(w, Z) = Q(w, (X, Y)) = \frac{1}{2}(w^T X - Y)^2 \tag{1}$$

where Z denotes the random variable from which the i.i.d data points $(z_i)_{i=1}^n = (x_i, y_i)_{i=1}^n$ are sampled. Similarly, X denotes the random variable corresponding to the x_i's and Y denotes the random variable corresponding to the y_i's such that $Z = (X, Y)$.

Also, let the dataset D be partitioned as $D = D_1 \cup D_2 \cup D_3$, such that D_1 represents the training data, D_2 represents the validation data, and D_3 represents the unseen test data.

The gradient of the loss function Q with respect to w is:

$$\frac{\partial Q}{\partial w} = X(w^T X - Y) \tag{2}$$

The training criterion C for the above problem is:

$$C = \frac{1}{2}\sum_{z_i \in D_1}(w^T x_i - y_i)^2 + \frac{1}{2}\sum_{j=1}^{d}\lambda_j w_j^2 \tag{3}$$

And the above C can be written in the following format:

$$C = a(\lambda) + b(\lambda)^T w + \frac{1}{2}w^T H(\lambda) w \tag{4}$$

where

$$a(\lambda) = \frac{1}{2}\sum_{z_i \in D_1} y_i^2 \text{ , } b(\lambda) = -\sum_{z_i \in D_1} y_i x_i \text{ and } H(\lambda) = \sum_{z_i \in D_1} x_i x_i^T + A(\lambda) \tag{5}$$

where $A = diag(\lambda_1, \lambda_2, ..., \lambda_d)$

To find the optimal parameters w for fixed hyperparameters λ, we can take the derivative of C with respect to w in equation (4) and equate it to zero as follows:

$$\frac{\partial C}{\partial w} = b(\lambda) + H(\lambda)w = 0 \tag{6}$$

Then we get the optimal parameters w as a function of λ:

$$w(\lambda) = -H^{-1}(\lambda)b(\lambda) \tag{7}$$

We define the validation error E as follows:

$$E(\lambda, D) = \frac{1}{|D_2|} \sum_{z_i \in D_2} Q(w(\lambda, D_1), z_i) \tag{8}$$

We write $w(\lambda, D_1)$ because w is a function of the hyperparameter λ and the training data D_1.

Differentiating E wrt w and then using equation (2), we have:

$$\frac{\partial E}{\partial w} = \frac{1}{|D_2|} \sum_{z_i \in D_2} \frac{\partial Q(w, z_i)}{\partial w} = \frac{1}{|D_2|} \sum_{z_i \in D_2} x_i(w^T x_i - y_i) \tag{9}$$

Now to optimize our λ, we can back-propagate our gradients through each operation required to solve the linear system in equation (7) as explained below.

Algorithm

Step 1: Take some initial value of $\lambda = \lambda_0$.

Step 2: From the training data D_1, we compute $H(\lambda) = \sum_{z_i \in D_1} x_i x_i^T + A(\lambda)$, as defined in equation (5). (We need not compute $\sum_i x_i x_i^T$ during every iteration, it's sufficient to compute it only once since the training data is not going to change).

Step 3: We find the Cholesky decomposition of $H(\lambda) = LL^T$ in $O(d^3)$ time. The lower triangular matrix L is computed as follows:

for i = 1,...,d
  $L_{i,i} = \sqrt{H_{i,i} - \sum_{k=1}^{i-1} L_{i,k}^2}$
  for j = i+1,...,d
    $L_{j,i} = (H_{i,j} - \sum_{k=1}^{i-1} L_{i,k} L_{j,k})/L_{i,i}$

Step 4: For a given λ, we can now find the optimal parameters w using the equation $-b = Hw$ (we get this equation after equating $\frac{\partial C}{\partial w}$ as 0). We solve this linear system in two steps using the Cholesky decomposition LL^T:

First solve $Lu = -b$ for u:

for i = 1,...,d
  $u_i = (-b_i - \sum_{k=1}^{i-1} L_{i,k} u_k)/L_{i,i}$

Then solve $L^T w = u$ for w:

for i = d,...,1
  $w_i = (u_i - \sum_{k=i+1}^{d} L_{k,i} w_k)/L_{i,i}$

Step 5: For each datapoint $(x_i, y_i) \in D_2$, we evaluate the loss incurred by the classifier, compute $\frac{\partial Q}{\partial w}$ using equation (2) and use this value to compute $\frac{\partial E}{\partial w}$ using equation (9). (Not taking into account the dependencies of w_i on w_j for $j > i$).

Step 6: Now our goal is to compute $\frac{\partial E}{\partial \lambda}$. In order to do that, first we compute the gradient of E with respect to H and b through the effect of H and b on w (as explained below). Since H is positive-definite and symmetric, we can use the Cholesky decomposition of H for backpropagating the gradients.

As intermediate results, the below algorithm computes the partial derivatives with respect to u and L as well as the "full gradients" with respect to w, $\frac{\partial E}{\partial w_i}|_{w_i}$, taking into account all the dependencies between the w_i's brought by the recursive computation of w_i's.

First backpropagate through the solution of $L^T w = u$:

```
initialise dEdw ← ∂E/∂w |_{w_1,...,w_d} (which we computed in step 5)
initialise dEdL ← 0
for i = 1,...,d
    dEdu_i ← dEdw_i / L_{i,i}
    dEdL_{i,i} ← dEdL_{i,i} − dEdw_i * w_i / L_{i,i}
    for k = i+1,...,d
        dEdw_k ← dEdw_k − dEdw_i * L_{k,i} / L_{i,i}
        dEdL_{k,i} ← dEdL_{k,i} − dEdw_i * w_k / L_{i,i}
```

Then backpropagate through the solution of $Lu = -b$ to obtain the gradient of E with respect to the coefficient $b(\lambda)$ of the training criterion, $\frac{\partial E}{\partial b_i}$, as well as with respect to the lower-diagonal matrix L, $\mathrm{dEdL}_{i,j} = \frac{\partial E}{\partial L_{i,j}}$.

```
for i = d,...,1
    ∂E/∂b_i ← −dEdu_i / L_{i,i}
    dEdL_{i,i} ← dEdL_{i,i} − dEdu_i * u_i / L_{i,i}
    for k = 1,...,i−1
        dEdu_k ← dEdu_k − dEdu_i L_{i,k} / L_{i,i}
        dEdL_{i,k} ← dEdL_{i,k} − dEdu_i * u_k / L_{i,i}
```

Finally we backpropagate through the Cholesky decomposition, to convert the gradients with respect to L into gradients with respect to the Hessian $H(\lambda)$ as explained below:

```
for i = d,...,1
    for j = d,..., i+1
        dEdL_{i,i} ← dEdL_{i,i} − dEdL_{j,i} L_{j,i} / L_{i,i}
        ∂E/∂H_{i,j} ← dEdL_{j,i} / L_{i,i}
        for k = 1,...,i−1
            dEdL_{i,k} ← dEdL_{i,k} − dEdL_{j,i} L_{j,k} / L_{i,i}
            dEdL_{j,k} ← dEdL_{j,k} − dEdL_{j,i} L_{i,k} / L_{i,i}
    ∂E/∂H_{i,i} ← (1/2)dEdL_{i,i} / L_{i,i}
    for k = 1,...,i−1
        dEdL_{i,k} ← dEdL_{i,k} − dEdL_{i,i} L_{i,k} / L_{i,i}
```

Since H is symmetric, we have only computed the gradients with respect to the diagonal and upper diagonal of H. Now we have finally computed $\frac{\partial E}{\partial H}$ and $\frac{\partial E}{\partial b}$.

Step 7: Now we use the functional form of $b(\lambda)$ and $H(\lambda)$ to compute gradient of E with respect to λ:

$$\frac{\partial E}{\partial \lambda} = \sum_{i=1}^{d} \frac{\partial E}{\partial b_i}\frac{\partial b_i}{\partial \lambda} + \sum_{i=1}^{d}\sum_{j=1}^{d} \frac{\partial E}{\partial H_{i,j}}\frac{\partial H_{i,j}}{\partial \lambda} \tag{10}$$

From equation (5), we derived:

$$a(\lambda) = \frac{1}{2}\sum_{z_i \in D_1} y_i^2 \ , \ b(\lambda) = -\sum_{z_i \in D_1} y_i x_i \text{ and } H(\lambda) = \sum_{z_i \in D_1} x_i x_i^T + A(\lambda) \tag{11}$$

where $A = diag(\lambda_1, \lambda_2, ..., \lambda_d)$

Differentiating $a(\lambda), b(\lambda)$ and $H(\lambda)$ with respect to λ, we get:

$$\frac{\partial a}{\partial \lambda_k} = 0, \frac{\partial b_i}{\partial \lambda_k} = 0 \text{ and } \frac{\partial H_{i,j}}{\partial \lambda_k} = \delta_{i,j}\delta_{j,k} \ \forall i, j, k \tag{12}$$

where $\delta_{i,j} = 1$ if $i = j$, otherwise 0

Plugging in the values of $\frac{\partial b_i}{\partial \lambda}$ and $\frac{\partial H_{i,j}}{\partial \lambda}$ into equation (10), we get the gradient of validation error E with respect to λ: $\frac{\partial E}{\partial \lambda}$.

Step 8: We improve our λ using gradient descent, using a suitable step-size α:

$$\lambda \leftarrow \lambda - \alpha\frac{\partial E}{\partial \lambda} \tag{13}$$

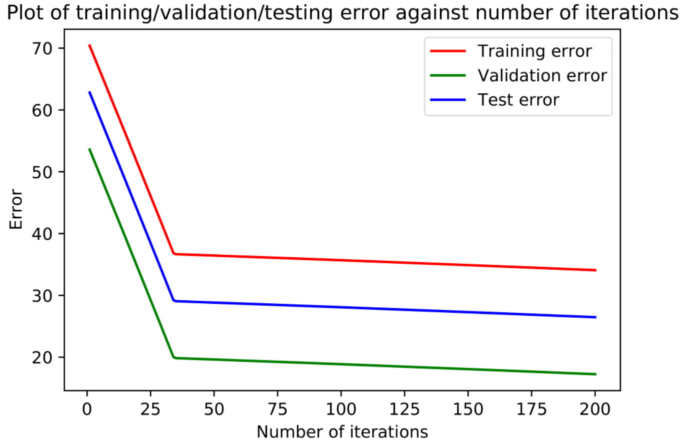
If the stopping condition is reached, we terminate the algorithm, else we repeat from step 2 of the algorithm.

Step 9: We report the training, validation and testing error for the parameter w and hyperparameter λ.

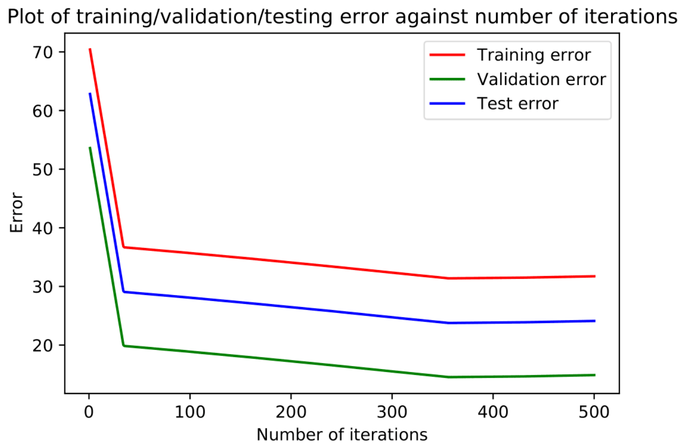
Results

We used the above hyperparameter optimization algorithm to learn a linear regressor for the 'median price prediction' problem and the corresponding 13-dimensional Boston Housing dataset. This hyperparameter optimization algorithm enables us to learn different regularization factors for each feature in the 13-dimensional dataset. The results are as follows:

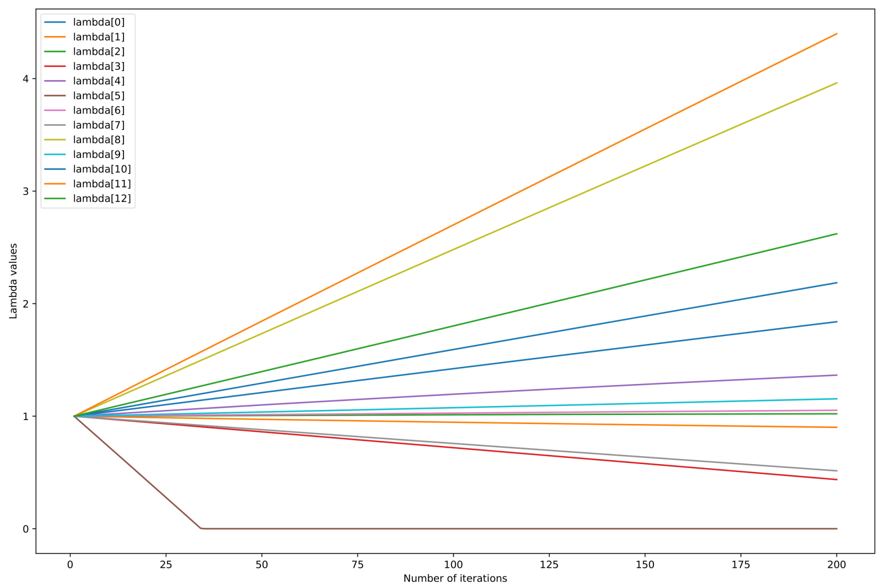
1. Plot of training, validation and testing error for 200 iterations:



2. Plot of training, validation and testing error for 500 iterations:



3. Plot of hyperparameters (λ's) when all of them are initialised to 1:



4. Plot of hyperparameters (λ's) when all of them are initialised randomly: