Regularization

Prof. Dr. Josef F. Bürgler

Studiengang Informatik
Hochschule Luzern, Informatik

I.BA_ML

Introduction

Topic: Regularization

Goals: Understand the concept of regularization in machine learning as a means to avoid overfitting.

Results: After these lectures You can judge Your model whether it overfits or underfits the data. You are able to take appropriate action to have a model which is just right.

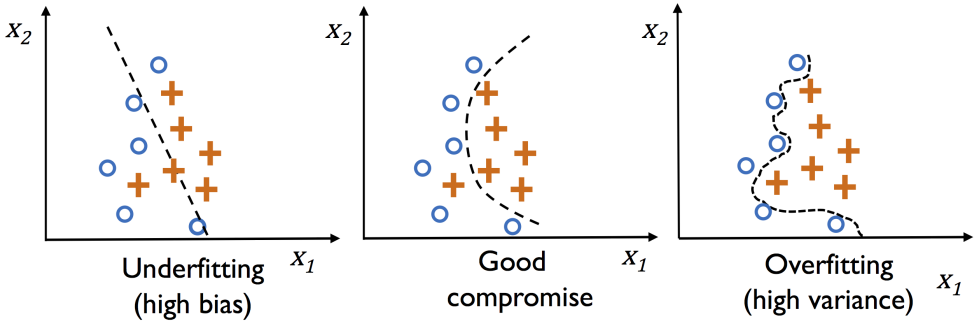
Further steps: We will first show, what we mean by under- and overfitting. Then we will show how ridge, LASSO and elastic regularization can be used to avoid the problem. Finally we will briefly mention the cross validation technique to detect and avoid under- or overfitting.

Content

Content

Under- and Overfitting

- **Underfitting** (also called high bias) means, that the model is not complex enough to capture the pattern in the training data well and therefore suffers from low performance on test (or unseen) data.
- **Overfitting** is a common problem in ML, where a model performs well on training data but does not generalize well on test (or unseen) data. We also say that the model has **high variance** which means, that a small change in the training data changes the parameters of the model drastically.

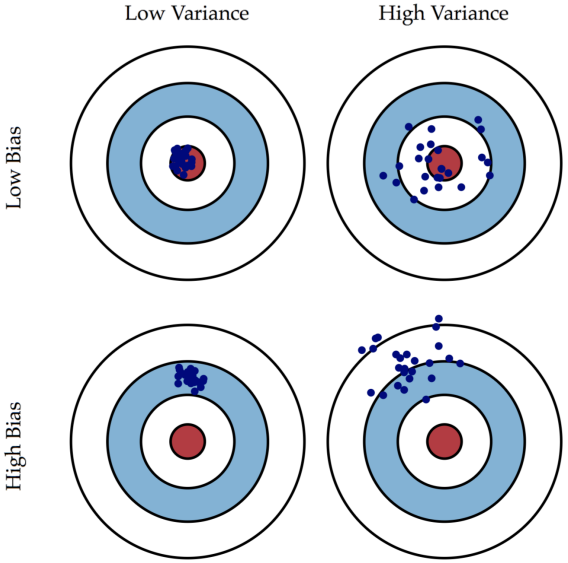


(Source [1])

High/Low Bias and High/Low Variance

This is yet another way to visualize high versus low bias or variance

- **Low Bias:** Average value is on center, i.e. correct.
- **High Bias:** Average value is far off, i.e. not correct.
- **Low Variance:** values do not spread.
- **High Bias:** values are spreading heavily.



(Source: http://scott.fortmann-roe.com/docs/BiasVariance.html)

Addressing overfitting

Possible options:

1. Reduce the number of features (parameters)
   - Manually select which features to keep
   - Model selection algorithm (further down)
2. Regularization
   - Keep all features, but reduce magnitude (values) of parameters θ_i.
   - Works well when we have lot's of features, each of which contribute just a bit to predicting y.

Example

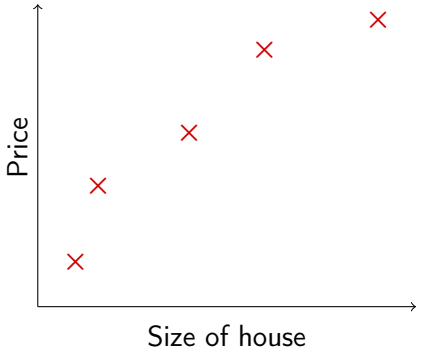
Let's say our (polynomial) regression models for the price of a house are

$$M_1 : \ h(\boldsymbol{\theta}, \boldsymbol{x}) = \boldsymbol{x}^\mathsf{T}\boldsymbol{\theta} \ = \ \theta_0 + \theta_1 x + \theta_2 x^2 \quad \text{and}$$
$$M_2 : \ h(\boldsymbol{\theta}, \boldsymbol{x}) = \boldsymbol{x}^\mathsf{T}\boldsymbol{\theta} \ = \ \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4.$$

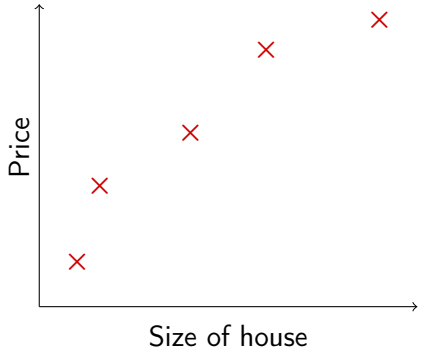
And we are given a training set of 5 houses (just as an unrealistic example).

Addressing overfitting (cont.)

Example



$M_1 : \theta_0 + \theta_1 x + \theta_2 x^2$



$M_2 : \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Suppose we penalize θ_3 and θ_4 forcing them to be small:

$$\min_{\boldsymbol{\theta}} \frac{1}{2n} \sum_{i=1}^{n} \left(h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)}) - y^{(i)} \right)^2 + 1000 \cdot \theta_3^2 + 1000 \cdot \theta_4^2$$

Addressing overfitting (cont.)

In **Ridge Regularization:** Use the following cost function

$$J(\boldsymbol{\theta}) = \frac{1}{2n}\left[\sum_{i=1}^{n}\left(h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)}) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{m} \theta_i^2\right]$$

Note:

- The second sum goes from 1 to the number of features m and it **does not contain** θ_0.
- Regularization parameters are $\theta_1, \theta_2, \ldots, \theta_n$.
- the first sum is the usual one and goes from 1 to the number of samples n.
- The **regularization hyperparameter** λ controls the amount of regularization. If $\lambda = 0$ there will be no regularization, and if $\lambda = \infty$ there will be underfitting because only θ_0 will be different from zero.

In (ridge) regularized linear regression, we choose $\boldsymbol{\theta}$ to minimize $J(\boldsymbol{\theta})$:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\boldsymbol{\theta})$$

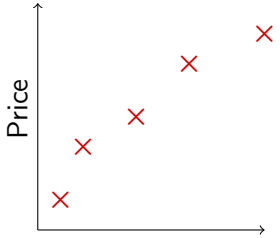
Content

High/Low Bias and High/Low Variance

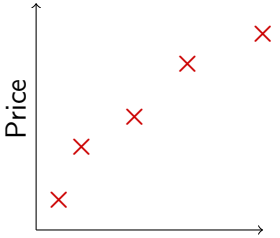
Linear regression with L^2- (or ridge) regularization:

- Model $h(\boldsymbol{\theta}, \boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{\theta} = \sum_{j=0}^{m} x_j\theta_j = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_m x_m$.
- Cost function

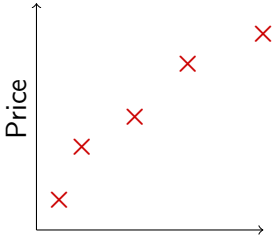
$$J(\boldsymbol{\theta}) = \frac{1}{2n}\left[\sum_{i=1}^{n}\left(h(\boldsymbol{\theta}, \boldsymbol{x}^{(i)}) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{m}\theta_j^2\right].$$



Large λ ($\approx$ 10′000); $\theta_i \approx 0$, $i = 1, 2, \ldots$; $h(\boldsymbol{\theta}, \boldsymbol{x}) \approx \theta_0$, $\rightarrow$ high bias (underfit)!



Intermediate λ; just right!



Small λ; high variance (overfit)!

General Procedure: Parameter Norm Penalties

- Introduce norm penalty $\Omega(\boldsymbol{\theta})$ which penalizes large values of θ.
- Use **regularized cost (objective) function**

$$\tilde{J}(\boldsymbol{\theta}; \mathbf{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \mathbf{X}, \boldsymbol{y}) + \lambda \Omega(\boldsymbol{\theta})$$

- The most common norm penalty $\Omega(\boldsymbol{\theta})$ is

$$\Omega_R(\theta) = \frac{1}{2n}\left(\theta_1^2 + \theta_2^2 + \cdots + \theta_m^2\right) = \frac{1}{2n}\sum_{k=1}^{m}\theta_k^2$$

  This is called L^2 **parameter (or ridge) regularization** because it tries to reduce the L^2 norm of the parameter vector $\boldsymbol{\theta}$.
- In L^1 **parameter (or LASSO [a]) regularization**

$$\Omega_L(\theta) = \frac{1}{n}\left(|\theta_1| + |\theta_2| + \cdots + |\theta_m|\right) = \frac{1}{n}\sum_{k=1}^{m}|\theta_k|$$

  In this case, the L^1 norm of the parameter vector $\boldsymbol{\theta}$ is reduced.

[a] LASSO: **L**east **A**bsolute **S**hrinkage and **S**election **O**perator

General Procedure: Elastic Regularization

In the **elastic regularization** we use a linear combination of the ridge and LASSO regularization:

$$\tilde{J}(\boldsymbol{\theta}; \mathbf{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \mathbf{X}, \boldsymbol{y}) + \lambda \left[r\Omega_L(\boldsymbol{\theta}) + (1-r)\Omega_R(\boldsymbol{\theta}) \right]$$

where

$$\Omega_L(\boldsymbol{\theta}) = \frac{1}{n} \sum_{k=1}^{m} |\theta_k|$$

and

$$\Omega_R(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{k=1}^{m} \theta_k^2$$

are the LASSO and ridge regularization term, respectively

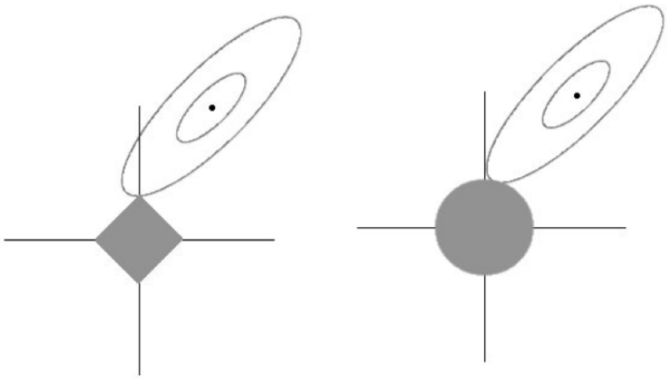
In the case $r = 0$ we have pure ridge regularization and in the case $r = 1$ we have pure LASSO regularization.

LASSO's feature/predictor selector

LASSO regularization has a feature called **predictor selector**. The figure shows contours of the error and constraint regions for the LASSO (left) and Ridge regression (right).

Isolines of the cost function are ellipses with the minimum at the center. LASSO only allows paramter vectors inside the diamond shape: this implies that one or more of the components of the parameter will be zero.

This will not be the case in the case of ridge regularization.



Gradient Descent

If we want to use the gradient descent method, we have to compute the gradient of the cost function $\tilde{J}(\boldsymbol{\theta}; \mathbf{X}, \boldsymbol{y})$:

$$\frac{\partial}{\partial \boldsymbol{\theta}} \tilde{J}(\boldsymbol{\theta}; \mathbf{X}, \boldsymbol{y}) = \frac{\partial}{\partial \boldsymbol{\theta}} J(\boldsymbol{\theta}; \mathbf{X}, \boldsymbol{y}) + \lambda \frac{\partial}{\partial \boldsymbol{\theta}} \Omega(\boldsymbol{\theta})$$

We already know the first term from previous sessions. In the case of ridge regularization $\Omega = \Omega_R$ we have

$$\frac{\partial}{\partial \boldsymbol{\theta}} \Omega_R(\boldsymbol{\theta}) = \frac{1}{n} \frac{\partial}{\partial \boldsymbol{\theta}} \sum_{j=1}^{m} \theta_j^2 = \frac{1}{n} [0, \theta_1, \theta_2, \theta_3, \ldots, \theta_m]^T$$

whereas in LASSO regularization $\Omega = \Omega_L$ it is

$$\frac{\partial}{\partial \boldsymbol{\theta}} \Omega_L(\boldsymbol{\theta}) = \frac{1}{n} \frac{\partial}{\partial \boldsymbol{\theta}} \sum_{j=1}^{m} |\theta_j| = \frac{1}{n} \left[0, \frac{|\theta_1|}{\theta_1}, \frac{|\theta_2|}{\theta_2}, \frac{|\theta_3|}{\theta_3}, \ldots, \frac{|\theta_m|}{\theta_m}\right]^T$$

Note: $\frac{d}{dx}|x| = \frac{|x|}{x}$ and we set $\frac{|\theta_i|}{\theta_i} = 0$ for any $\theta_i = 0$.

Gradient Descent (cont.)

In the case of ridge regularization:

Repeat (until convergence) {

$$\theta_0 = \theta_0 - \alpha \frac{1}{n} \left[\sum_{i=1}^{n} \left(h(\boldsymbol{\theta}_k, \boldsymbol{x}^{(i)}) - y^{(i)} \right) \boldsymbol{x}^{(i)} \right]$$

$$\theta_j = \theta_j - \alpha \frac{1}{n} \left[\sum_{i=1}^{n} \left(h(\boldsymbol{\theta}_k, \boldsymbol{x}^{(i)}) - y^{(i)} \right) \boldsymbol{x}^{(i)} + \lambda \theta_j \right], \quad j = 1, 2, \ldots, m$$

}

In the case of LASSO regularization:

Repeat (until convergence) {

$$\theta_0 = \theta_0 - \alpha \frac{1}{n} \left[\sum_{i=1}^{n} \left(h(\boldsymbol{\theta}_k, \boldsymbol{x}^{(i)}) - y^{(i)} \right) \boldsymbol{x}^{(i)} \right]$$

$$\theta_j = \theta_j - \alpha \frac{1}{n} \left[\sum_{i=1}^{n} \left(h(\boldsymbol{\theta}_k, \boldsymbol{x}^{(i)}) - y^{(i)} \right) \boldsymbol{x}^{(i)} + \lambda \frac{|\theta_j|}{\theta_j} \right], \quad j = 1, 2, \ldots, m$$

}

Regularization and normal equations

- In linear regression the matrix $\mathbf{X}^\mathsf{T}\mathbf{X}$ has to be inverted if using the normal equations:

$$\boldsymbol{\theta} = \left(\mathbf{X}^\mathsf{T}\mathbf{X}\right)^{-1}\mathbf{X}^\mathsf{T}\boldsymbol{y}$$

- The matrix $\mathbf{X}^\mathsf{T}\mathbf{X}$ might be singular as in the case $\mathrm{n} < \mathrm{m}$ (less samples than features) or if some features are highly correlated.
- Regularization leads to following normal equations

$$\boldsymbol{\theta} = \left(\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda \begin{bmatrix} 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}\right)^{-1} \mathbf{X}^\mathsf{T}\boldsymbol{y}$$

Note the 11-element of the matrix multiplied by λ is zero!

Regularization and normal equations (cont.)

Remember the data matrix $\mathbf{X}$ and the target vector $\boldsymbol{y}$:

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \cdots & x_m^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \cdots & x_m^{(2)} \\ 1 & x_1^{(3)} & x_2^{(3)} & x_3^{(3)} & \cdots & x_m^{(3)} \\ 1 & x_1^{(4)} & x_2^{(4)} & x_3^{(4)} & \cdots & x_m^{(4)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & x_2^{(n)} & x_3^{(n)} & \cdots & x_m^{(n)} \end{bmatrix} \quad \text{and} \quad \boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ y^{(4)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

The i-th sample is $(\boldsymbol{x}^{(i)}, y^{(i)})$ where $\boldsymbol{x}^{(i)} = \left[x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, \ldots, x_m^{(1)}\right]$, which is the i-th row of the data matrix except the element in the first row.

Typically $n >> m$, i.e. the number of samples is much larger than the number of features.

How to choose the regularization hyperparameter λ

Split data into

- training set (60%), $(\boldsymbol{x}^{(i)}, y^{(i)})$, $i = 1, \ldots, n$
- cross validation (cv) set (20%), $(\boldsymbol{x}_{cv}^{(i)}, y_{cv}^{(i)})$, $i = 1, \ldots, n_{cv}$
- test set (20%), $(\boldsymbol{x}_{t}^{(i)}, y_{t}^{(i)})$, $i = 1, \ldots, n_{t}$

For each value of the regularization hyperparameter λ compute parameter vector $\boldsymbol{\theta}$ by minimizing the training error $J_{train}(\boldsymbol{\theta})$.

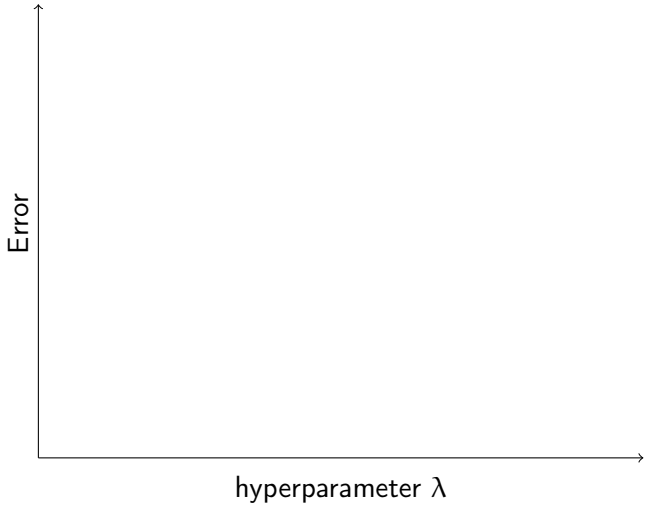
1. Trail 1: $\lambda = 0 \rightarrow \min_{\boldsymbol{\theta}} J_{train}(\boldsymbol{\theta}) \rightarrow \boldsymbol{\theta}^{(1)} \rightarrow J_{CV}(\boldsymbol{\theta}^{(1)})$
2. Trail 2: $\lambda = 0.01 \rightarrow \min_{\boldsymbol{\theta}} J_{train}(\boldsymbol{\theta}) \rightarrow \boldsymbol{\theta}^{(2)} \rightarrow J_{CV}(\boldsymbol{\theta}^{(2)})$
3. Trail 3: $\lambda = 0.02 \rightarrow \min_{\boldsymbol{\theta}} J_{train}(\boldsymbol{\theta}) \rightarrow \boldsymbol{\theta}^{(3)} \rightarrow J_{CV}(\boldsymbol{\theta}^{(3)})$
4. Trail 4: $\lambda = 0.04 \rightarrow \min_{\boldsymbol{\theta}} J_{train}(\boldsymbol{\theta}) \rightarrow \boldsymbol{\theta}^{(4)} \rightarrow J_{CV}(\boldsymbol{\theta}^{(4)})$
5. Trail 5: $\lambda = 0.08 \rightarrow \min_{\boldsymbol{\theta}} J_{train}(\boldsymbol{\theta}) \rightarrow \boldsymbol{\theta}^{(5)} \rightarrow J_{CV}(\boldsymbol{\theta}^{(5)})$
6. $\vdots$
7. Trail 12: $\lambda = 10.24 \rightarrow \min_{\boldsymbol{\theta}} J_{train}(\boldsymbol{\theta}) \rightarrow \boldsymbol{\theta}^{(12)} \rightarrow J_{CV}(\boldsymbol{\theta}^{(12)})$

How to choose the regularization hyperparameter λ

For each value of the regularization hyperparameter λ draw the error of the training set $J_{\text{train}}(\boldsymbol{\theta}^{(\lambda)})$ and the the error in the cross validatation set, $J_{CV}(\boldsymbol{\theta}^{(\lambda)})$.

Choose λ such that the cross validation error $J_{CV}(\boldsymbol{\theta}^{(\lambda)})$ is smallest, i.e.

$$\hat{\lambda} = \underset{\lambda}{\text{argmin}}\, J_{CV}(\boldsymbol{\theta}^{(\lambda)})$$



Content

How to choose the right model

Suppose we want to decide which of the polynomial regression models

$$M_k : \ h(\boldsymbol{\theta}, \boldsymbol{x}) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_k x^k, \quad k = 1, 2, \ldots, 10$$

we should choose.

How can we automatically select k that represents a good tradeoff between bais and variance?

On first sight, the following might be a good algorithm

- For each $k = 1, 2, \ldots, 10$ train the model M_k on a given training set S to get $\boldsymbol{\theta}_k$.
- Pick the k with the smallest training error.

This algorithm does not work! The higher the order of the polynomial, the better it will fit the training set S and thus the lower the training error. This method will therefor select the high-variance, high-degree polynomial, which is often a poor chioce.

Hold-out (or Simple) Cross Validation

The following algorithm, called **hold-out, or simple cross validation** is better

1. Randomly split S into S_{train} (say, 70% of the data) and S_{cv} (the remaining 30%), called the the **hold-out cross validation set**.

2. For each k (degree of the polynomial), train the model M_k on S_{train} to get $\boldsymbol{\theta}_k$, i.e the parameter vector for the polynomial of degree k

3. Select the $\boldsymbol{\theta}_k$ that had the smallest error $\hat{\varepsilon}_{S_{\text{cv}}}(\boldsymbol{\theta}_k)$, i.e.

$$k = \underset{k \in \{1,2,\ldots,10\}}{\operatorname{argmin}} \hat{\varepsilon}_{S_{\text{cv}}}(\boldsymbol{\theta}_k)$$

   where $\hat{\varepsilon}_{S_{\text{cv}}}(\boldsymbol{\theta}_k)$ is the empirical error, if we use $\boldsymbol{\theta}_k$ as the parameter vector on the set of examples in S_{cv}.

By testing on S_{cv}, i.e. on a set of examples the model was not trained on, we obtain a better estimate of the true generalization error of a particular parameter vector.

Problem of the hold-out cross validation data set: it wastes about 30% of the data.

k-fold Cross Validation

If we don't want to waste 30% of data, we could hold out less data each time: we us **k-fold cross validation**

1. Randomly split S into l disjoint subsets $S_1, S_2, \ldots, S_l$ of n/l training examples each.

2. Evaluate each model M_k, $k \in \{1, 2, \ldots, 10\}$, (every polynomial with degree from 1 to 10) as follows:
   - For $j = 1, 2, \ldots, l$ train model M_k on all data, except S_j to get $\boldsymbol{\theta}_{kj}$ and test $\boldsymbol{\theta}_{kj}$ on S_j to get the empirical error $\hat{\varepsilon}_{S_j}(\boldsymbol{\theta}_{kj})$ if we use $\boldsymbol{\theta}_{kj}$ as the parameter vector in M_k.
   - The estimated generalization error of model M_k is then calculated as the average of the $\hat{\varepsilon}_{S_j}(\boldsymbol{\theta}_{kj})$'s, i.e.

$$\frac{1}{l}\sum_{j=1}^{l}\hat{\varepsilon}_{S_j}(\boldsymbol{\theta}_{kj}) \quad \text{where} \quad \hat{\varepsilon}_{S_j}(\boldsymbol{\theta}_{kj}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h(\boldsymbol{\theta}_{kj}, \boldsymbol{x}^{(i)}) - y^{(i)}\right)^2$$

3. Pick the model M_k with the lowest estimated generalization error and retrain that model on the entiry training set S. The resulting hypothesis is the final answer.

Conclusion

- We now know what regularization does in regression.
- We understand how regularization can be used to avoid the problem of overfitting.
- We know, that too much regularization leads to underfitting.
- We understand, can implement and apply various regularization methods (ridge, LASSO, elastic).
- We have learned a method to choose the proper regularization hyperparameter.
- We have learned crucial points of the cross validation technique.

I'm happy to answer Your

Questions

References I

[1] Sebastian Raschka and Vahid Mirjalili. *Python Machine Learning, 2nd Ed.*. 2nd ed. Birmingham, UK: Packt Publishing, 2017. ISBN: 978-1787125933.