Machine Learning Cheatsheet

Edoardo Riggio

June 7, 2022

Machine Learning - S.P. 2022
Computer Science
Università della Svizzera Italiana, Lugano

Contents

1 Introduction

What does it mean "to learn"? We have to different definitions, one from a **statistical perspective**, and one from a **computer science perspective**.

- **Statistical Perspective**

  Vast amounts of data are being generated in many fields. The statistician's job is to make sense of all of this data, extract meaningful patterns and trends, and understand "what the data says". This approach is also known as **learning from data**.

- **Computer Science Perspective**

  The field of machine learning is concerned with how to construct computer programs that automatically improve with experience.

1.1 Mitchell's Formalisation

A computer program is said to learn from **experience** E – concerning some class of **task** T, and **performance measurement** P – if its performance at task T, as measured by P, improves with experience E.

1.2 Models

1.2.1 White Box Model

In this case, both physical laws and structural parameters of the problem are known. A family equation can be derived.

1.2.2 Grey Box Model

The physical laws are known in this case, and at least one parameter is unknown. A family of equations can be derived, but the parameters need to be identified.

1.2.3 Black Box Model

In this case, the physical laws are unknown. A family of equations cannot be derived.

1.3 Measures and Measurements

The operation of measuring an unknown quantity x_0 can be modeled as taking an instance – i.e., a **measurement** – x_i at time i, with an ad-hoc sensor S.

Although S has been suitably designed and realized, the physical elements that compose it are far from ideal and introduce uncertainties in the measurement process. As a result, x_i only represents an estimate of x_0.

1.4 Types of Models

1.4.1 Additive Model

The measurement process can be modeled as:

$$x = x_0 + \eta \qquad \text{where } \eta = f_n(0, \sigma_\eta^2)$$

Where η is an independent and identically distributed random variable, the model assumes that the i.i.d. noise does not depend on the working point x_0.

1.5 Multiplicative Model

The measurement process can be modeled as:

$$x = x_0(1 + \eta) \qquad \text{where } \eta = f_n(0, \sigma_\eta^2)$$

Where η is an independent and identically distributed random variable, the noise, in this case, depends on the working point x_0. In absolute terms, the impact of the noise on the signal is $x_0\eta$, but the relative contribution is η – which does not depend on x_0.

1.6 Supervised Learning

In a supervised learning framework, we have the following elements: a **concept to learn**, a **teacher**, and a **student**.

1.6.1 Regression

The goal of regression is to determine the function that explains the given instances – **measuremets**. The student proposes a family of models $f(\theta, x)$, and after a learning procedure, the "best" model $f(\hat{\theta}, x)$ is found.

1.6.2 Classification

The goal of classification is to determine the function – **model** – that partitions the input – **measurements** – into classes. The student proposes a family of models $f(\theta, x)$, and after a learning procedure, the "best" model $f(\hat{\theta}, x)$ is found.

1.6.3 Prediction

The goal of prediction is to tell us which data – **measurements** – will come next, possibly along with a confidence level. The student proposes a family of models $f(\theta, x)$, and after a learning procedure, the "best" model $f(\hat{\theta}, x)$ is found.

1.7 Features

We might want to extract features from the measurements to ease the learning task. The features must:

- Provide a compact representation of inputs
- Be particularly advantageous if we have prior information to take advantage of
- Be reduced to a minimal set before processing them for task solving

1.8 Unsupervised Learning

The goal of unsupervised learning is to build a representation of data. During its operational life, given an input, the machine provides information that can be used for decision-making.

2 Linear Regression

To prepare a linear regression model, several techniques can be used. The most popular being the **Least Mean Square (LMS)**, and the **Gradient Descent**.

Linear models are good when we have the following conditions:

- The data is generated by a linear model
- The dataset is small
- The data is sparse
- The uncertainty is high

2.1 Muliple Linear Regression

We are given a set of points:

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \qquad \text{where } x \in \mathbb{R}^d, y \in \mathbb{R}$$

This is known as the **training set**. We now assume that the unknown function that generates the data is linear. Moreover, we assume that there is a gaussian uncertainty affecting the measurements. The model is given as:

$$y(x) = \theta_1^0 + \theta_2^0 z_2 + \cdots + \theta_d^0 z_d \quad \text{where } \theta^0 \in \mathbb{R}, \eta = N(0, \sigma_\eta^2)$$

Which can be written in its canonical form:

$$y(x) = x^T \theta^0 + \eta \quad \text{where } x^T = \begin{bmatrix} 1 & z_2 & \cdots & z_d \end{bmatrix}, {\theta^0}^T = \begin{bmatrix} \theta_1^0 & \theta_2^0 & \cdots & \theta_d^0 \end{bmatrix}$$

Both the optimal parameters and the variance of the noise are unknown. Since we know that the system model is linear, the family of models which best fits the data is:

$$\hat{y}(x) = f(\theta, x) = x^T\theta$$

In order to determine the best parameters, we estimate them:

$$f(\hat{\theta}, x) = x^T\hat{\theta}$$

2.1.1 Least Mean Square

The main idea of this procedure is to select the linear function that minimizes the average distance between the given points and the linear function.

The **performance function** of this method is the following:

$$V_n(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y(x_i) - f(\theta, x_i))^2$$

Therefore, the parameter vector $\hat{\theta}$ minimizing the performance function is the following:

$$\hat{\theta} = \arg\min_{\theta\in\Theta} V_n(\theta)$$

2.1.2 Parameter Estimation

By grouping the data into vectors X – of all x values – and Y – of all y values, we can rewrite the formula above in its canonical form:

$$\begin{aligned} V_n^*(\theta) &= \sum_{i=1}^{n}(y(x_i) - x_i^T\theta)^2 \\ &= (Y - X\theta)^T(Y - X\theta) \end{aligned}$$

Stationary points are those for which:

$$\frac{\partial V_n^*(\theta)}{\partial\theta} = -2X^TY + 2X^TX\theta = 0$$

Therefore, the parameter vector $\hat{\theta}$ minimizing the performance function is the following:

$$\hat{\theta} = (X_TX)^{-1}X^TY$$

And the best approximating model is:

$$f(\hat{\theta}x) = x^T\hat{\theta}$$

2.2 Performance at Task

In order to test the model, we need to use another set of unseen data. This is because the **performance at task**:

$$V_n(\hat{\theta}) = \frac{1}{n}\sum_{i=1}^{n}(y(x_i) - f(\theta, x_i))^2$$

It is biased to the training set. For this reason, we consider another set of data – different from the training set – and call it **test set**:

$$\{(\bar{x}_1, \bar{y}_1), (\bar{x}_2, \bar{y}_2), \ldots, (\bar{x}_l, \bar{y}_l)\}$$

And evaluate the performance at task on it:

$$V_l(\hat{\theta}) = \frac{1}{l}\sum_{i=1}^{l}(\bar{y}_i - \bar{x}^T\hat{\theta})^2$$

2.3 Cross-Validation

Cross-validation provides a means to assess the performance of a model. This method can also be considered for model selection – with some care. In the latter case, we consider the model that minimizes

$$V_l(\hat{\theta}) = \frac{1}{l}\sum_{i=1}^{l}(\bar{y}_i - \bar{x}^T\hat{\theta})^2$$

2.4 Properties

Under the linear framework presented earlier, it can be proved that both implications hold true:

$$\lim_{n\to\infty} \hat{\theta} = \theta^0$$

$$\lim_{l\to\infty} V_l(\hat{\theta}) = \sigma_\eta^2$$

In addition, assuming that n training couples are given, then it can be proved that:

$$Var(\hat{\theta}) = (X^TX)^{-1}\sigma_\eta^2 \qquad \text{with } \hat{\sigma}_\eta^2 = \frac{1}{n-d}\sum_{i=1}^{n}(y(x_i) - f(\hat{\theta}, x_i))^2$$

If a parameter is smaller than twice its standard deviation, it must be set to 0. After which, we re-evaluate the performance and decide whether to keep it. This is known as **Occam's razor strategy**.

2.5 Ridge Regression

Ridge regression aims at pushing as many parameters as possible towards zero. This is done by adding a shrinking penalty to the loss function. The **MSE** training performance mesure

$$V_n(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i^T\theta)^2$$

Now becomes

$$V_{Ridge}(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i^T\theta)^2 + \lambda\sum_{i=2}^{d}\theta_i^2$$

Where λ is a hyperparameter weighting the two contributions. A smaller λ gives more importance to accuracy, while a high λ priviledges a smaller number of parameters in the model. A tradeoff can be obtained by estimating and appropriate λ thanks to the **validation set**.

2.6 Lasso Regression

Lasso regression also aims at pushing as many parameters as possible towards zero. In this case, however, we penalize the parameter itself rather than its square. The **MSE** now is:

$$V_{Lasso}(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i^T\theta)^2 + \lambda\sum_{i=2}^{d}|\theta_i|$$

The optimization problem is not convex anymore, and the loss function is not differentiable. This makes the problem computationally complex and solvable only via optimization techniques such as quadratic programming.

2.7 Final Prediction Error

The expected prediction error is computed as follows:

$$FPE = \frac{n+d}{n-d}\sum_{i=1}^{n}(y(x_i) - f(\hat{\theta}, x_i))^2$$

This measurement can be used for both model selection – only on hierarchical family of models, and for assessing the performance of unseen data.

3 Non-Linear Regression

To determine the stationary points of a differentiable function we use the following minimization problem:

$$\hat{\theta} = \arg\min_{\theta\in\Theta} V_n(\theta)$$

This minimization is carried out by a **gradient descent-based procedure**.

3.1 Gradient-Based Optimization

Given a generic convex and differentiable scalar function $y = f(x)$, we do the following to determine the unique stationary point.

We start from an initial point $x(0) = x_0$ and start moving along the gradient in the direction minimizing the loss function. To do so we use the following formula:

$$\theta_{i+1} = \theta_i - \varepsilon_L \frac{\partial V_n(\theta)}{\partial \theta}|_{\theta_i}$$

The minimization procedure applied to a non-linear function – and based on a training set – is also known as **learning procedure**.

When minimizing, we might encounter what is known as the **identifiability problem**. This happens when the function to be optimized is not convex, and there are several local or global minima in the function.

3.2 Structural Risk

The structural risk of the estimated model is its generalization ability. To compute the function's generalization ability, we use the following formula:

$$\bar{V}(\hat{\theta}) = \int L(y, f(\hat{\theta}, x)) p_{x,y} dxy$$

Where L is the **loss function**. The risk associated with the model's generalization ability can be decomposed into three terms:

$$\bar{V}(\hat{\theta}) = (\bar{V}(\hat{\theta}) - \bar{V}(\theta^0)) + (\bar{V}(\theta^0) - V_I) + V_I$$

3.2.1 Inherent Risk

The inherent risk depends entirely on the structure of the learning problem. This term can be improved only by improving the problem itself, such as reducing the noise caused by the measurement instruments.

The inherent risk is computed as:

$$V_I$$

3.2.2 Approximation Risk

The approximation risk depends on how close the approximating family of models is to the function generating the data ($g(x)$). We can use a more appropriate family of models to reduce this risk.

The approximation risk is computed as:

$$\bar{V}(\theta^0) - V_I$$

3.2.3 Estimation Risk

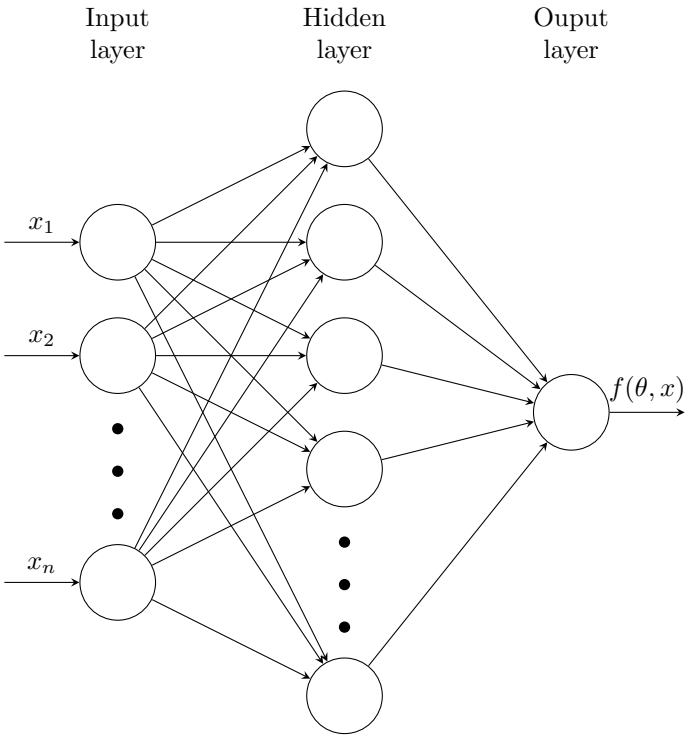
The estimation risk depends on the effectiveness of the learning procedure. We can use a more effective learning procedure to reduce this risk.

The estimation risk is computed as:

$$\bar{V}(\hat{\theta}) - \bar{V}(\theta^0)$$

4 Feedforward Neural Networks

A neural network has the following structure



Each connection between neurons has a weight of w_n. Moreover, the overall weight of a sigle neuron is computed as:

$$a_t = \sum_{i=1}^{n} x^i w^i$$

Each neuron in the hidden layer has an activation function. This activation function can be of several different types:

- **Sigmoidal**

  This activation function is mainly used by the neurons of the hidden layer. The sigmoidal formula is:

$$Sig(x) = \frac{1}{1 + e * -x}$$

- **Hyperbolic Tangent**

  This activation function is an alternative to the sigmoidal activation function. This function is useful when we have back-propagation. The hyperbolic tangent formula is:

$$HT(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

- **Linear**

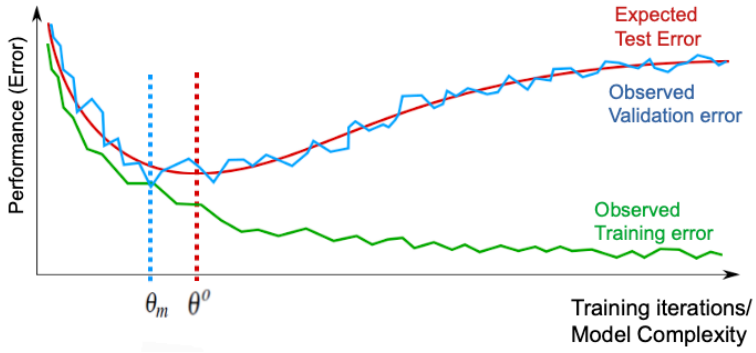
  The neurons of the output layer mainly use this activation function.

4.1 Universal Approximation Theorem

A feedforward neural network with a single hidden layer containing a finite number of neurons and a linear output neuron approximates any continuous function defined on compact subsets.

4.2 Early Stopping

The idea of early stopping is to start from an over-dimensioned network and try to find where the models start diverging.



To evaluate the model, we use another set of data called **validation set**. We cannot use the test set for the model selection during training. Otherwise, the model performance assessment would be biased.

4.3 Splitting the Data

To split the data we have – which has length n, we apply a ratio (which is application-dependent). This ratio is computed in the function of:

- The size of n
- The complexity of the model

For example:

$$n = 1,000 \rightarrow Tr = 70\%,\ V = 15\%,\ Te = 15\%$$

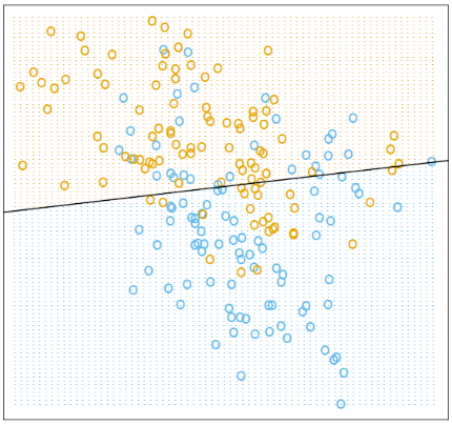
$$n = 1,000,000 \rightarrow Tr = 99\%,\ V = 0.5\%,\ Te = 0.5\%$$

5 Classification Problem

Given an input vector x, and a set of classes $C_1, C_2, \ldots, C_k$, we wish to assign x to the most appropriate class. In classification, y assumes a categorical value.

5.1 Binary Classifier

Given an input vector x of two features and two classes, we divide the features in a binary manner – i.e., either it is in one group or the other.



Given a training set, we obtain the following estimated parameter:

$$\hat{\theta} = (X^T X)^{-1} X^T X$$

And given a new x, we evaluate the following discriminant function:

$$f(\hat{\theta}, x) = x^T \hat{\theta}$$

If the discriminant value is above 0.5, then x will be part of one group. Otherwise, x will be part of the other group.

5.2 Bayes Classifier

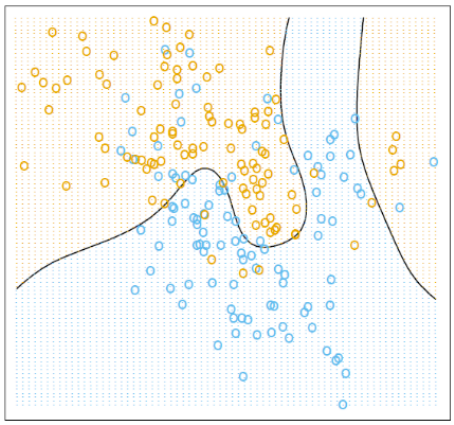
The Bayes classifier deals with multi-class problems. Here we assign a probability to each class and select the one that maximizes such probability.

$$Pr(Y = k|X = x)$$

Where X and Y are two random variables. The Bayes theorem described above provides us with a way to express the posterior conditional probability as:

$$Pr(Y = k|X = x) = \frac{Pr(X = x|Y = k) \cdot Pr(Y = k)}{Pr(X = x)}$$

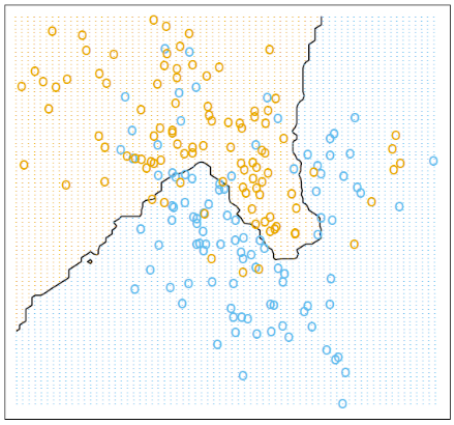
Where $Pr(Y = k|X = x)$ is the **posterior probability**, $Pr(X = x|Y = k)$ is the **likelihood**, $Pr(Y = k)$ is the **prior probability**, and $Pr(X = x)$ is the evidence.



This classifier is said to be naïve. It is because it makes the strong assumption of independence among the features. The Bayes classifier is optimal if the premises are met and the known probabilities.

5.3 K-Nearest Neighbour Classifier

This type of classifier looks at the k-nearest neighbors of point x to make the classification decision. For example, the Euclidean distance can be used to determine the class.



In this case, no training phase is required, but we have a high computational cost in evaluating the distances.

5.4 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a robust way to solve the classification problem and find a separating boundary.

Here we reformulate the Bayes theorem in terms of probability density functions. Thus:

$$Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}$$

Where $\pi_k = Pr(Y = k)$ is the **prior probability**, and $f_k(x)$ represents the **likelihood of the k-th class** and is defined as:

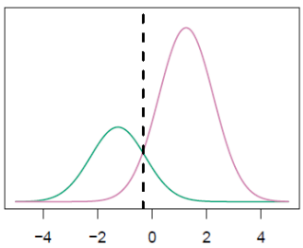
$$f_k(x) = Pr(X = x|Y = k)$$

5.4.1 One-Dimensional Setting

If we consider a one-dimensional setting and Gaussian distributed classes:

$$f_k(x) = \frac{1}{\sqrt{2\pi} \cdot \sigma_k} e^{-\frac{1}{2}\left(\frac{x-\mu_k}{\sigma_k}\right)^2}$$

Which can be represented graphically as:



In the case of a scalar x and identical variances, the Bayes formula becomes:

$$Pr(Y = k|X = x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\cdot\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu_k}{\sigma}\right)^2}}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\cdot\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu_l}{\sigma}\right)^2}}$$

To obtain the line that separates the classes, we are looking for the maximum of the above probability. We simplify the formula by taking the log and removing the constant terms. The discriminant function now becomes:

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

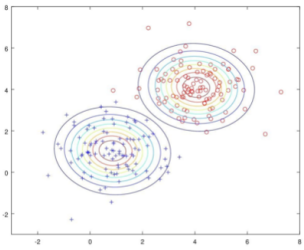
The decision boundary between two classes satisfies the equation $\delta_k(x) = \delta_i(x)$.

5.4.2 Vector Space Setting

In the case of vectors, we will have the following formula:

$$f(x) = \frac{1}{(2\pi)^{p/2}|\sum|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \sum^{-1}(x-\mu)}$$

Which can be represented graphically as:



Now the linea discriminant will become:

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$$

With two classes, the LDA coincides with the linear regression classification method. Here, the decision boundary between two classes satisfies the equation $\delta_k(x) = \delta_i(x)$.

5.5 Perceptron

The perceptron algorithm was invented to solve pattern classification problems. This method works well on linearly separable classes, and its parameters can be learned.

The architecture of a perceptron is that of a neural network with a single neuron with a *Heaviside* activation function. After a random assignment of weights, these are iteratively updated during the training procedure.

$$w_{t+1}^j = w_t^j - \varepsilon_L(y_i - f(w_t, x_i))x_i^j$$

The algorithm will converge to the optimal parameter – if the problem is linearly separable.

5.6 Classification with Feedforward Neural Networks

Given a performance function:

$$V_n(\theta) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(\theta, x_i)$$

We determine the parameter estimate by solving the following minimization problem:

$$\hat{\theta} = \arg\min_{\theta\in\Theta} V_n(\theta)$$

This is carried out by the following learning procedure:

$$\theta_{i+1} = \theta_i - \varepsilon_L \frac{\partial V_n(\theta)}{\partial\theta}\Big|_{\theta=\theta_i}$$

And finally, obtain the classifier:

$$f(\hat{\theta}, x)$$

5.7 Logistic Regression

Linear classifiers derived as extensions of the regression methods neither provide a bounded output nor a probabilistic interpretation.

Logistic regression aims at training the network's parameters so that a probabilistic network supports the sigmoidal output.

6 Model Performance

6.1 Quality Assessment if the Solution

There are several different ways to estimate the performance of a classifier.

6.1.1 Apparent Error Rate

This method computes the empirical risk to estimate the structural one. The set Z_n is used to infer the model and estimate its accuracy performance.

AER is a strongly optimistically biased estimate unless n is very large.

6.1.2 Crossvalidation

Crossvalidation estimates the generalization error $\bar{V}(\hat{\theta})$ on a new dataset. The sets S_D and S_E are generated by **randomly** splitting Z_n into two disjoint subsets. we use S_D as the **training set**, and S_E as the **test set**.

This method can be considered an unbiased estimate if S_E is large enough.

6.1.3 K-Fold Crossvalidation

This method randomly splits Z_n into k disjoint subsets of equal size. For each subset, the other $k-1$ remaining subsets are merged to form S_D, and the k subset is used as S_E. The resulting k estimates are averaged.

6.1.4 Leave-One-Out

In this method, S_E contains one pattern of Z_n, and S_D contains the remaining $n-1$ patterns of Z_n. The procedure iterates n times every time, holding out each pattern in Z_n. The resulting n estimates are averaged.

This is a simplification of K-CV. The estimates from each fold are highly correlated. Thus, the estimate has a significant variance.

6.1.5 Bootstrap Method

If we have a very small dataset, we use this method. The algorithm is similar to Leave-One-Out. Although, in this case, patterns are selected from Z_n **with replacement**.

This method underestimates the test error $\bar{V}(\hat{\theta})$

6.2 Model Validity

Model validity is used to define which model is the best one.

6.3 Test on Residuals

The verification of the hypothesis $E[\varepsilon] = 0$ requires a test on the sample mean. The null hypothesis we consider is $H_0 : E[\varepsilon] = 0$, while the alternate hypothesis is $H_1 : E[\varepsilon] \neq 0$. To see which hypothesis holds, we need to design a statistic.

Under the **null hypothesis** H_0, the Central Limit Theorem grants the T-Student statistic to follow a normal distribution. We compute T as such:

$$T = \frac{\bar{\varepsilon}}{\sqrt{\frac{s^2}{l}}} \sim N(0,1)$$

$$\bar{\varepsilon} = \frac{1}{l} \sum_{i=1}^{l} \varepsilon_i$$

$$s^2 = \frac{1}{l} \sum_{i=1}^{l} \varepsilon_i^2$$

If T is outside of the 95% confidence interval – i.e., [-1.96, 1.96] – then we reject the null hypothesis H_0.

Something similar can be done to check if one model is statistically better than the other. In this case we consider the null hypothesis to be:

$$H_0 : Var[\varepsilon_a] = Var[\varepsilon_b]$$

$$H_1 : Var[\varepsilon_a] \neq Var[\varepsilon_b]$$

If T – or rather a slightly modified version of it – is out of the 95% confidence interval, we reject the null hypothesis. This means that the two models are different. Now that we know that they are different, we compute their variances to find out which model is the best. The model with the lowest variance is the best one.