Supervised Learning
Basic principles. Regression

Alexander Trofimov
PhD, professor, NRNU MEPhI

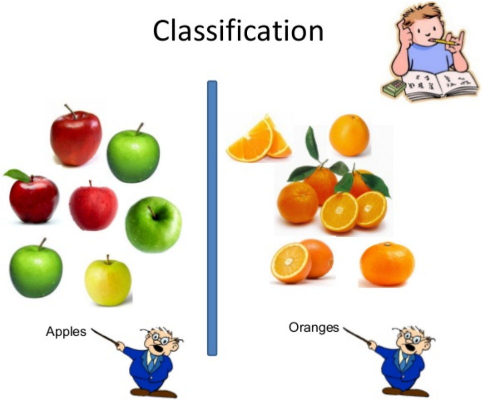
lab@neuroinfo.ru
http://datalearning.ru

Course "Machine Learning"

November 2022

Objects and Responses

$\mathscr{X}$ — instance domain
$\mathscr{Y}$ — response domain
$F : \mathscr{X} \to \mathscr{Y}$ — unknown mapping (target function)



Features

$f_j : \mathscr{X} \to D_j$ — j-th feature
D_j — j-th feature domain, $\quad j = 1, ..., M$

Types of features:

- $D_j = \{0, 1\}$ — binary feature f_j
- $|D_j| < \infty$ — nominal (categorical) feature f_j
- $|D_j| < \infty$, D_j is ordered — ordinal feature f_j
- $D_j \subseteq \mathbb{R}$ — real-valued feature f_j

$x \in \mathscr{X}$ — some object from $\mathscr{X}$

$f(x) = (f_1(x), ..., f_M(x))$ — feature vector of object x

$f(x) \in D_1 \times ... \times D_M$

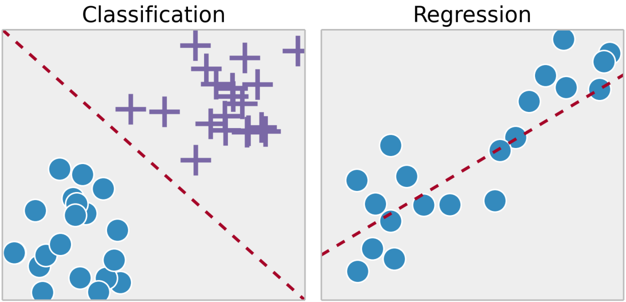
Types of Responses

Regression:

- $\mathscr{Y} = \mathbb{R}$ or $\mathscr{Y} = \mathbb{R}^L$

Classification:

- $\mathscr{Y} = \{-1, 1\}$ or $\mathscr{Y} = \{0, 1\}$ — binary classification
- $\mathscr{Y} = \{1, ..., K\}$ — multiclass classification



Supervised Learning. Problem Statement

$\mathscr{X}$ — instance domain
$\mathscr{Y}$ — response domain
$F : \mathscr{X} \to \mathscr{Y}$ — target function (unknown)

Given:
$\mathscr{D} = \{(x^{(1)}, y^{(1)}), ..., (x^{(n)}, y^{(n)})\}$ — available data sample
$x^{(1)}, ..., x^{(n)} \in \mathscr{X}$ — set of instances
$y^{(1)}, ..., y^{(n)} \in \mathscr{Y}$ — set of responses
$y^{(i)} = F(x^{(i)}), \quad i = 1, ..., n$

Find out:
$h : \mathscr{X} \to \mathscr{Y}$ — estimation of F (hypothesis)

Questions:

- What does “estimation” mean?
- How to construct h?

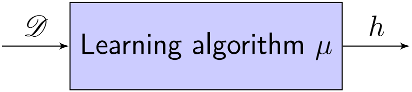
Learning Algorithm

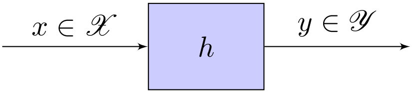
Definition

Learning algorithm μ is a mapping of arbitrary data sample $\mathscr{D} \in (\mathscr{X} \times \mathscr{Y})^n$ to hypothesis $h \in \mathscr{H}$:

$$\mu : (\mathscr{X} \times \mathscr{Y})^n \to \mathscr{H},$$

where $\mathscr{H}$ is a given domain in functional space $\mathscr{X} \to \mathscr{Y}$ (hypothesis domain, class of models)





Examples of hypotheses domains $\mathscr{H}$:

$\mathscr{H} = \{h : h(x) = \sum_{j=1}^{M} \beta_j f_j(x)\}, \quad \mathscr{Y} = \mathbb{R}$

$\mathscr{H} = \{h : h(x) = sign \sum_{j=1}^{M} \beta_j f_j(x)\}, \quad \mathscr{Y} = \{-1, 1\}$

Inductive Learning

The aim of machine learning is rarely to replicate the data from $\mathscr{D}$ but the prediction for new cases

Induction is inference from particular cases to the general case

The hypothesis h is inductive because it is assumed to approximate F well over unseen examples (even though it is only derived from the given data sample)

Why the model learned on the data from $\mathscr{D}$ will predict responses for new cases accurately?

Learning algorithm μ solves ill-posed problem where the data by itself is not sufficient to reconstruct the target function F

So because learning is ill-posed we should make some extra assumptions to have a unique solution with the data we have

Inductive Bias

Definition

Inductive bias of the learning algorithm is the set of assumptions that makes ill-posed machine learning problem to have unique solution

Inductive bias is not to be confused with statistical bias. Unlike statistical bias, which is a numerical value, inductive bias is a set of assumptions

We introduce inductive bias when we assume a class of hypotheses $\mathscr{H}$ that can be generated by learning algorithm. That is to say, we define a family of models but let the data determine which of these models is most appropriate

Inductive Bias. Examples

Too weak inductive bias:

- Weak assumptions about class $\mathscr{H}$
- Huge family of models
- Much sensitivity to the data
- Many degrees of freedom

Too strong inductive bias:

- Strong assumptions about $\mathscr{H}$
- No flexibility in the model
- Ignoring the data
- Few degrees of freedom
- "Bad" bias leads to inadequate model

Examples of inductive bias:

- for classification: cases that are near each other tend to belong to the same class, distinct classes tend to be separated by wide boundaries
- for regression: regression function is linear

Generalization, Underfitting and Overfitting

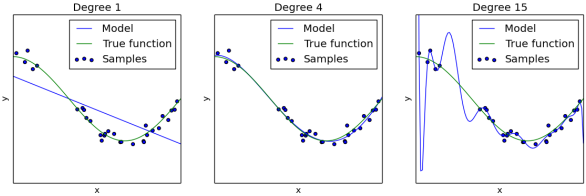
How to measure the quality of inductive bias?

Definition

Generalization of the model is its ability to accurately predict responses for previously unseen data

Underfitting: model cannot capture the underlying trend or patterns in the data

Overfitting: model describes random error or noise instead of the underlying relationship



Estimating the Generalization

How to measure generalization of a model?

Generalization ability of a model is related to the quality of its inductive bias

To measure the generalization we need unseen data

Training data is used to fit the model

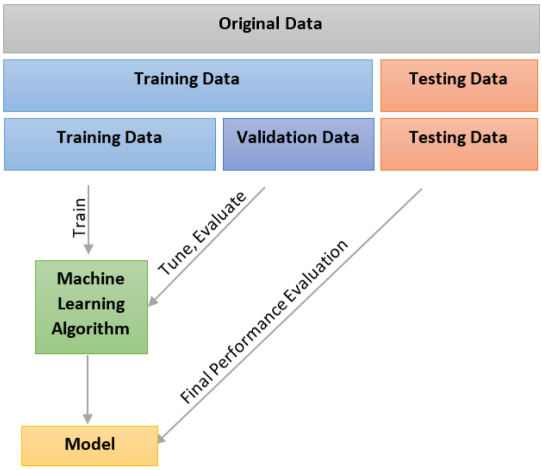
Validation data is used to test the generalization ability of the trained models and select the best one

Test data is used to final accuracy estimation of the model

$$\mathscr{D} = (\mathscr{D}_T \cup \mathscr{D}_V) \cup \mathscr{D}_{Tst}$$

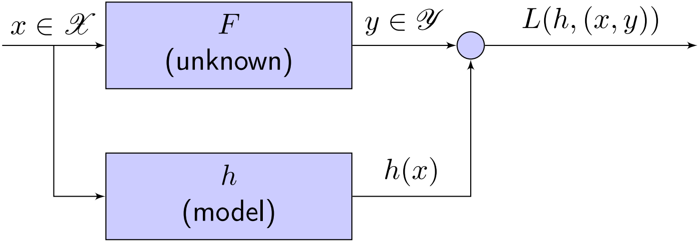
$$\mathscr{D}_{Tr} = \mathscr{D}_T \cup \mathscr{D}_V$$

Training, Validation and Test Samples



Validation sample is considered to be a part of the training process

Loss Function



Definition

Loss function (cost function) $L(h, (x, y)) \in \mathbb{R}^+$ is some measure of predictive inaccuracy of model $h \in \mathscr{H}$ at $(x, y) \in \mathscr{X} \times \mathscr{Y}$

When comparing the same type of loss among many models, lower loss indicates a better model

The best value: $L(h, (x, y)) = 0$ (means no error on $x \in \mathscr{X}$)

Statistical View

The map $F : \mathscr{X} \to \mathscr{Y}$ usually is not deterministic

Suppose that y is an observation of random variable Y with conditional distribution $f_Y(y|x)$ for a given $x \in \mathscr{X}$

Suppose that vector $x \in \mathscr{X}$ is a random vector drawn from distribution $f_X(x)$

Hence, the loss function $L(h, (X, Y))$ is a random variable (as a function of random vector (X, Y))

$\mathscr{D} = \{(x^{(1)}, y^{(1)}), ..., (x^{(n)}, y^{(n)})\}$ is a sample drawn from joint probability distribution $f_{XY}(x, y) = f_Y(y|x)f_X(x)$

In practice, $f_{XY}(x, y)$ is usually unknown

Risk and Empirical Risk

Definition

Risk $R(h)$ associated with model h is expectation of the loss function:

$$R(h) = \mathrm{M}[L(h,(X,Y))] = \int_{\mathscr{X}\times\mathscr{Y}} L(h,(x,y)) f_{XY}(x,y)dxdy$$

Definition

Empirical risk $R^*(h)$ associated with model h is an estimation of risk $R(h)$ as mean value of the loss function over sample $\mathscr{D}$:

$$R^*(h) = \frac{1}{n}\sum_{i=1}^{n} L\left(h, \left(x^{(i)}, y^{(i)}\right)\right)$$

ERM principle

Empirical risk $R^*(h)$ represents a error of model h over sample $\mathscr{D}$

Example for regression tasks:

$L(h,(x,y)) = (h(x) - y)^2$ — quadratic loss function

$R^*(h) = \frac{1}{n}\sum_{i=1}^{n}\left(h(x^{(i)}) - y^{(i)}\right)^2$ — mean squared error (MSE)

Example for classification tasks:

$L(h,(x,y)) = [h(x) \neq y]$ — 0-1 loss function

$R^*(h) = \frac{1}{n}\sum_{i=1}^{n}\left[h(x^{(i)}) \neq y^{(i)}\right]$ — classification error

Objective of learning algorithm μ:

$R_T^*(h) \to \min\limits_{h \in \mathscr{H}}$ — empirical risk minimization (ERM)

The learning algorithm defined by the ERM principle consists in solving this optimization problem

Risk for Quadratic Loss

$h(x)$ — response of the model h at given $x \in \mathscr{X}$ (determined)
$Y = F(x)$ — value of target function at given $x \in \mathscr{X}$ (random)

Risk for quadratic loss function $L(h, (x, Y))$ at given $x \in \mathscr{X}$:

$$r(h, x) = \mathrm{M}[L(h, (x, Y))|x] = \mathrm{M}\left[(h(x) - Y)^2|x\right]$$
$$= h^2(x) - 2h(x)\mathrm{M}[Y|x] + \mathrm{M}[Y^2|x]$$

Risk for Quadratic Loss

$h(x)$ — response of the model h at given $x \in \mathscr{X}$ (determined)
$Y = F(x)$ — value of target function at given $x \in \mathscr{X}$ (random)

Risk for quadratic loss function $L(h, (x, Y))$ at given $x \in \mathscr{X}$:

$$
\begin{aligned}
r(h, x) &= \mathrm{M}[L(h, (x, Y))|x] = \mathrm{M}\left[(h(x) - Y)^2|x\right] \\
&= h^2(x) - 2h(x)\mathrm{M}[Y|x] + \mathrm{M}[Y^2|x] \\
&= h^2(x) - 2h(x)\mathrm{M}[Y|x] + \mathrm{D}[Y|x] + (\mathrm{M}[Y|x])^2
\end{aligned}
$$

Risk for Quadratic Loss

$h(x)$ — response of the model h at given $x \in \mathscr{X}$ (determined)
$Y = F(x)$ — value of target function at given $x \in \mathscr{X}$ (random)

Risk for quadratic loss function $L(h,(x,Y))$ at given $x \in \mathscr{X}$:

$$\begin{aligned} r(h,x) &= \mathrm{M}[L(h,(x,Y))|x] = \mathrm{M}\left[(h(x)-Y)^2|x\right] \\ &= h^2(x) - 2h(x)\mathrm{M}[Y|x] + \mathrm{M}[Y^2|x] \\ &= h^2(x) - 2h(x)\mathrm{M}[Y|x] + \mathrm{D}[Y|x] + (\mathrm{M}[Y|x])^2 \\ &= (h(x)-\mathrm{M}[Y|x])^2 + \sigma_x^2 \end{aligned}$$

$(h(x)-\mathrm{M}[Y|x])^2$ — error of model h at given $x \in \mathscr{X}$

$\sigma_x^2 = \mathrm{D}[Y|x]$ — noise, doesn't depend on $\mathscr{D}$ or h

$h(x) = \mathrm{M}[Y|x], \forall x \in \mathscr{X} \Leftrightarrow h(x)$ is a regression function y on x

Bias-Variance Decomposition

$r(h, x) = (h(x) - \mathrm{M}[Y|x])^2 + \sigma_x^2$ — risk at given $x \in \mathscr{X}$

Hypothesis h formed by learning algorithm μ depends on training data $\mathscr{D}_T$: $h(x, \mathscr{D}_T)$ — response of the model h at given $x \in \mathscr{X}$ trained on random sample $\mathscr{D}_T$

Expectation over all random samples $\mathscr{D}_T$:

$$\mathrm{M}\left[(h(x, \mathscr{D}_T) - \mathrm{M}[Y|x])^2\right]$$
$$= \mathrm{M}\left[h(x, \mathscr{D}_T)^2\right] - 2\mathrm{M}[h(x, \mathscr{D}_T)]\mathrm{M}[Y|x] + \mathrm{M}[Y|x]^2$$

Bias-Variance Decomposition

$r(h, x) = (h(x) - \mathrm{M}[Y|x])^2 + \sigma_x^2$ — risk at given $x \in \mathscr{X}$

Hypothesis h formed by learning algorithm μ depends on training data $\mathscr{D}_T$: $h(x, \mathscr{D}_T)$ — response of the model h at given $x \in \mathscr{X}$ trained on random sample $\mathscr{D}_T$

Expectation over all random samples $\mathscr{D}_T$:

$$\begin{aligned} &\mathrm{M}\left[(h(x, \mathscr{D}_T) - \mathrm{M}[Y|x])^2\right] \\ &= \mathrm{M}\left[h(x, \mathscr{D}_T)^2\right] - 2\mathrm{M}[h(x, \mathscr{D}_T)]\mathrm{M}[Y|x] + \mathrm{M}[Y|x]^2 \\ &= \mathrm{D}\left[h(x, \mathscr{D}_T)\right] + \mathrm{M}[h(x, \mathscr{D}_T)]^2 - 2\mathrm{M}[h(x, \mathscr{D}_T)]\mathrm{M}[Y|x] + \mathrm{M}[Y|x]^2 \end{aligned}$$

Bias-Variance Decomposition

$r(h, x) = (h(x) - \mathrm{M}[Y|x])^2 + \sigma_x^2$ — risk at given $x \in \mathscr{X}$

Hypothesis h formed by learning algorithm μ depends on training data $\mathscr{D}_T$: $h(x, \mathscr{D}_T)$ — response of the model h at given $x \in \mathscr{X}$ trained on random sample $\mathscr{D}_T$

Expectation over all random samples $\mathscr{D}_T$:

$$
\begin{aligned}
&\mathrm{M}\left[(h(x, \mathscr{D}_T) - \mathrm{M}[Y|x])^2\right] \\
&= \mathrm{M}\left[h(x, \mathscr{D}_T)^2\right] - 2\mathrm{M}[h(x, \mathscr{D}_T)]\mathrm{M}[Y|x] + \mathrm{M}[Y|x]^2 \\
&= \mathrm{D}\left[h(x, \mathscr{D}_T)\right] + \mathrm{M}[h(x, \mathscr{D}_T)]^2 - 2\mathrm{M}[h(x, \mathscr{D}_T)]\mathrm{M}[Y|x] + \mathrm{M}[Y|x]^2 \\
&= (\mathrm{M}[h(x, \mathscr{D}_T)] - \mathrm{M}[Y|x])^2 + \mathrm{D}\left[h(x, \mathscr{D}_T)\right]
\end{aligned}
$$

$(\mathrm{M}[h(x, \mathscr{D}_T)] - \mathrm{M}[Y|x])$ — statistical bias of model h at given x
$\mathrm{D}\left[h(x, \mathscr{D}_T)\right]$ — variance of model h over training samples $\mathscr{D}_T$

Bias-Variance Trade-off

Risk of the model h at given $x \in \mathscr{X}$ (expectation over training samples $\mathscr{D}_T$):

$$R(h,x) = \left(\mathrm{M}[h(x,\mathscr{D}_T)] - \mathrm{M}[Y|x]\right)^2 + \mathrm{D}\left[h(x,\mathscr{D}_T)\right] + \sigma_x^2$$

$$R(h,x) = Bias^2[h] + \mathrm{D}[h] + \sigma_x^2$$

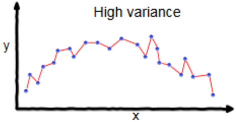
Three sources of error:

- $Bias^2[h]$ — error due to incorrect assumptions (bad inductive bias)
- $\mathrm{D}[h]$ — error due to variance of training samples (inability to perfectly estimate model's parameters from limited and noisy data)
- σ_x^2 — unavoidable error (doesn't depend on model)

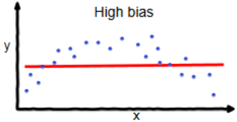
Inductive bias determines trade-off between bias and variance of model h

Bias-Variance Trade-off. Illustration 1

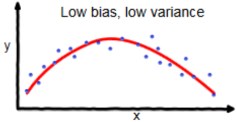
Overfitting



Underfitting



Good balance



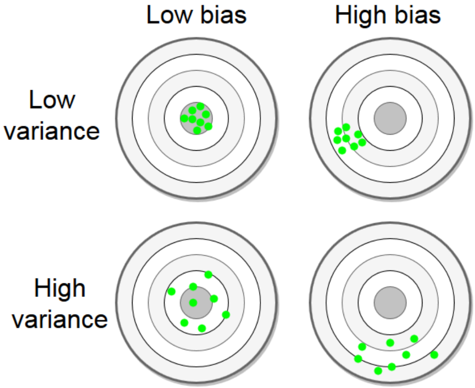
High variance leads to overfitting
High bias leads to underfitting

Strong inductive bias: low or high high bias, low variance
Weak inductive bias: low or high bias, high variance

Appropriate inductive bias leads to low bias, low variance

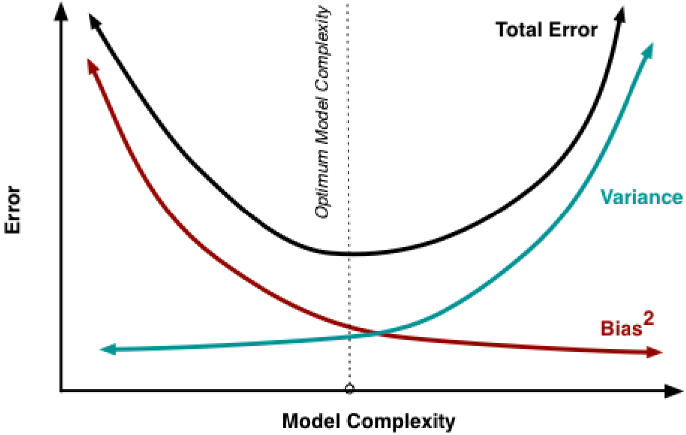
Bias-Variance Trade-off. Illustration 2

Dartboard = hypothesis space
Bullseye = target function
Darts = learned models



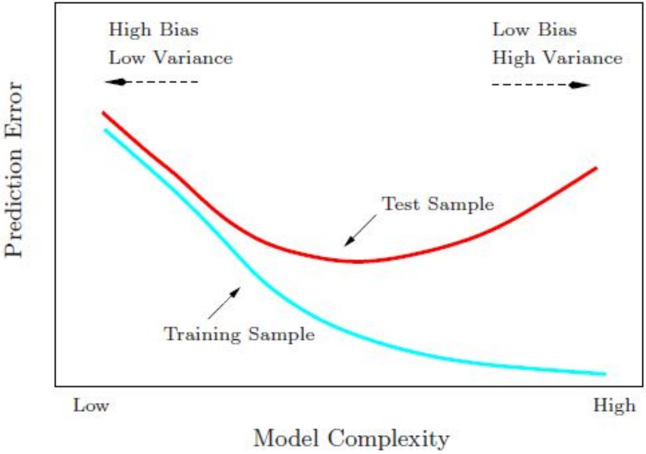
Inductive Bias and Model Complexity

Strong inductive bias leads to low model complexity
Weak inductive bias leads to high model complexity



Generalization and Model Complexity

Generalization of model depends on its complexity



Estimation of Model Error

$\mathscr{D} = (\mathscr{D}_T \cup \mathscr{D}_V) \cup \mathscr{D}_{Tst}$ — available data
$h \in \mathscr{H}$ — model trained on $\mathscr{D}_T$

How to estimate the error of the model h?

$R(h)$ — true risk of model h

$R^*_T(h), R^*_V(h), R^*_{Tst}(h)$ — empirical risks (e.g. MSE) over train, validation and test samples

$R^*_T(h)$ — this estimate is optimistic (i.e. biased)
$R^*_V(h)$ — was used in training process
$R^*_{Tst}(h)$ — estimation of model error over unseen examples

$R^*_{Tst}(h)$ looks good estimation but...

Estimation of Model Error

$\mathscr{D} = (\mathscr{D}_T \cup \mathscr{D}_V) \cup \mathscr{D}_{Tst}$ — available data
$h \in \mathscr{H}$ — model trained on $\mathscr{D}_T$

How to estimate the error of the model h?

$R(h)$ — true risk of model h

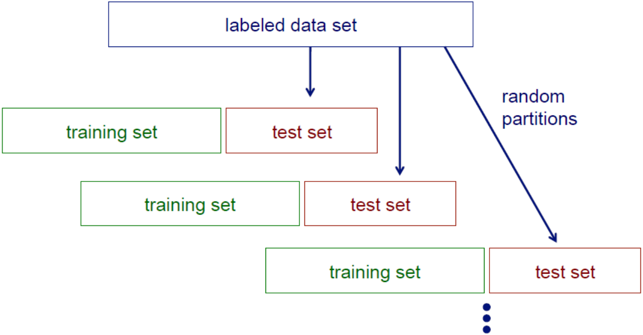
$R^*_T(h), R^*_V(h), R^*_{Tst}(h)$ — empirical risks (e.g. MSE) over train, validation and test samples

$R^*_T(h)$ — this estimate is optimistic (i.e. biased)
$R^*_V(h)$ — was used in training process
$R^*_{Tst}(h)$ — estimation of model error over unseen examples

$R^*_{Tst}(h)$ looks good estimation but... a single training and test set don't tell us how sensitive error is to a particular training sample

Partitioning the Data

Solution: repeatedly partitioning the available data into training and test sets



h_i — model trained on training data from i-th partition, $i = 1, ..., k$
$R^*_{Tst}(h_1), ..., R^*_{Tst}(h_k)$ — estimations of risk for models $h_1, ..., h_k$

Cross-Validation Techniques

Definition

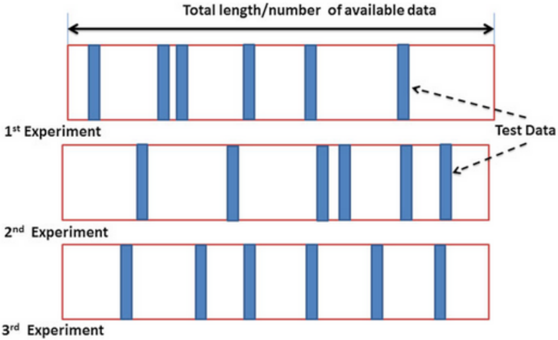
Cross-validation (CV) is a model evaluation technique used to assess a machine learning algorithm's performance in making predictions on new datasets that it has not been trained on

Cross validation techniques:

- Repeated random sub-sampling (Monte-Carlo CV)
- k-fold
- Holdout
- Leave-one-out (LOOCV)
- Resubstitution

Resubstitution does not partition the data, uses the training data for validation

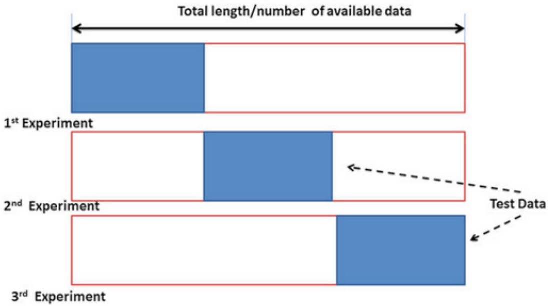
Repeated Random Sub-sampling CV



Whole data is randomly partitioned into training and test subsamples k times in specified proportion

Sample of errors: $R^*_{Tst}(h_1), ..., R^*_{Tst}(h_k)$

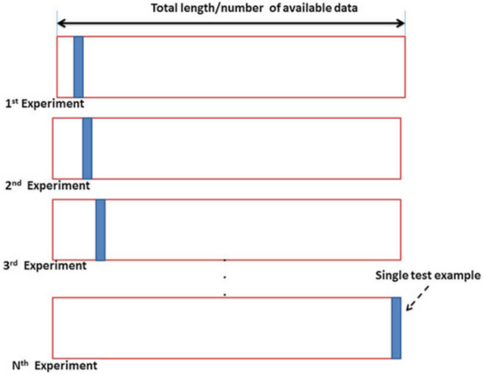
k-fold CV



Whole data is randomly partitioned into k equal sized subsamples (folds). One of k folds is retained as the test data, and the remaining $k-1$ folds are used as training data

Sample of errors: $R^*_{Tst}(h_1), ..., R^*_{Tst}(h_k)$

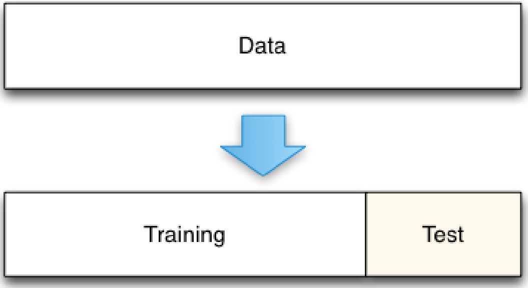
Leave-one-out CV



LOOCV is particular case of k-fold CV when $k = n$

Sample of errors: $R^*_{Tst}(h_1), ..., R^*_{Tst}(h_n)$

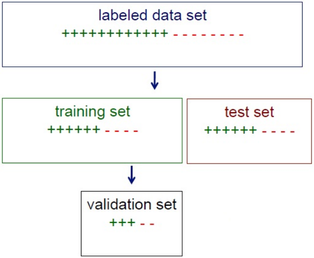
Holdout CV



Whole data is randomly partitioned into two sets: training and test subsamples in specified proportion

Sample of errors: $R^*_{Tst}(h)$

Stratified sampling



The test subsets (folds) are selected so that the mean response value is approximately equal in all the folds

In the case of a classification, stratified cross-validation keep the distribution of class labels in each fold

In practice: first stratify instances by class, then randomly select instances from each class proportionally

True error estimation

Whenever we use multiple training sets, as in k-fold CV and random sub-sampling CV, we are evaluating a learning algorithm μ, no individual learned model h

The true error R_{Tst} is the error when tested on the entire population of data instances

Sample of errors: $R^*_{Tst}(h_1), ..., R^*_{Tst}(h_k)$

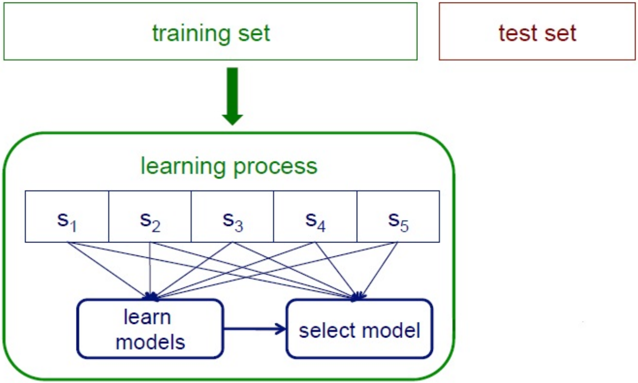
Point estimator: $\overline{R}_{Tst} = \frac{1}{k}\sum_{i=1}^{k} R^*_{Tst}(h_i)$

Variance: $s^2[R_{Tst}] = \frac{1}{k}\sum_{i=1}^{k}(R^*_{Tst}(h_i) - \overline{R}_{Tst})^2$

The cross-validation estimator $\overline{R}_{Tst}$ is very nearly unbiased for R_{Tst}
The variance $s^2[R_{Tst}]$ can be reduced by increasing the size of test set

Internal Cross-Validation

Instead of a single validation set, we can use cross-validation within a training set (e.g. to find meta-parameters and select a model)



Overview

- Inductive bias
- Generalization, underfitting and overfitting
- Training, validation and test samples
- Loss function, risk and empirical risk
- ERM principle
- Bias-variance decomposition
- Bias-variance trade-off
- Cross-validation techniques

Linear Regression Models. Problem Statement

Given:

$\mathscr{D}_T = \{(x^{(1)}, y^{(1)}), ..., (x^{(n)}, y^{(n)})\}$ — train data sample, $\mathscr{Y} = \mathbb{R}$

$f_1(x), ..., f_M(x)$ — features of object $x \in \mathscr{X}$

$\mathscr{H} = \{h : h(x) = \sum_{j=1}^{M} \beta_j f_j(x)\}$ — class of hypotheses (linear models)

$L(h, (x, y)) = (h(x) - y)^2$ — quadratic loss function

Objective:

Find parameters $\beta_1, ..., \beta_M$ that minimize empirical risk over train sample $\mathscr{D}_T$:

$$R^*(h) \to \min_{\beta_1, ..., \beta_M}$$

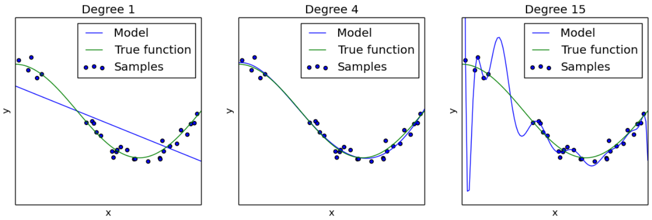
Linear Regression Models. Background

Empirical risk: $R^*(h) = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{M}\beta_j f_j(x^{(i)}) - y^{(i)}\right)^2$

Solution: $\beta = (F^T F)^{-1} F^T y$,

where $F = \begin{pmatrix} f_1(x^{(1)}) & \dots & f_M(x^{(1)}) \\ \dots & \dots & \dots \\ f_1(x^{(n)}) & \dots & f_M(x^{(n)}) \end{pmatrix}$ — design matrix,

$y = \left(y^{(1)}, \dots, y^{(n)}\right)^T$ — response vector



Validation of Regression Model

How to validate the trained model h?

The validation process involves:

- Analyzing the goodness-of-fit of the regression

  Coefficient of determination: $R^2 = 1 - \frac{D_{residual}}{D_{total}}$

- Analyzing the regression residuals

  Residual at $x^{(i)}$, $i = 1, ..., n$:

  $$e(x^{(i)}) = y^{(i)} - \sum_{j=1}^{M} \beta_j f_j(x^{(i)})$$

  - Graphical analysis
  - Quantitative analysis

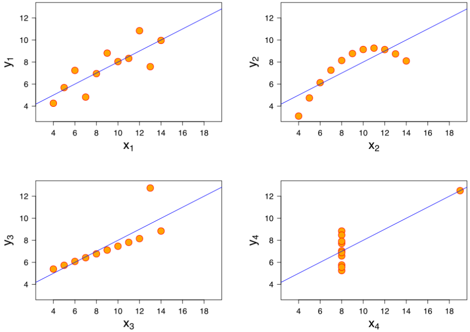
- Analyzing the regression performance on unseen data

  Empirical risk (MSE) on test set

Analyzing the Coefficient of Determination

Coefficient of determination R^2 close to 1 does not guarantee that the model fits the data well!

Anscombe's quartet



All four sets are identical when examined using simple summary statistics (mean, variance, correlation, linear regression, coefficient of determination), but vary considerably when graphed

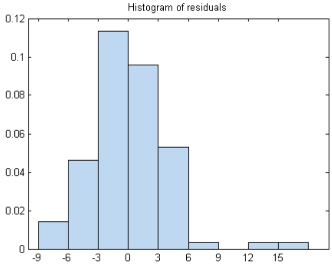
Graphical Analysis of Residuals

If the residuals appear to behave randomly, it suggests that the model should be adequate to the data

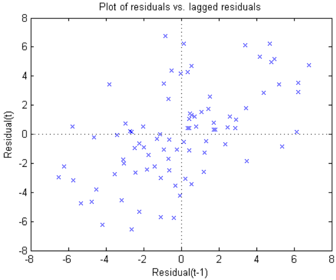
- Sufficiency of the functional part of the model: scatter plots of residuals versus predictors
- Non-constant variation across the data (heteroscedasticity: scatter plots of residuals versus predictors and fitted variable
- Independence of residuals: lag plot
- Normality of residuals: histogram

Plots of the residuals versus predictors or fitted variable that exhibit systematic structure indicate that the form of the modelled function can be improved in some way

Residual Analysis. Examples 1

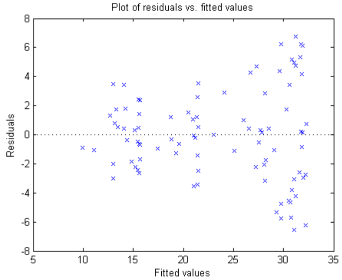


Some outliers appear
Exclude outliers and train new model



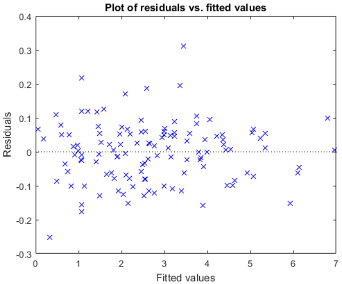
Many points in the upper-right and lower-left quadrants, indicating positive serial correlation (autocorrelation) among the residuals
Use autoregressive or other models

Residual Analysis. Examples 2



There is some tendency for larger fitted values to have larger residuals. Perhaps the model errors are proportional to the measured values

Use some variance-stabilizing transformations of variables, use other loss function



There is no obvious patterns, model seems to be adequate

Quantitative Analysis of Residuals

Types of analyses:

- Sufficiency of the functional part of the model
  — lack-of-fit tests
- Test for heteroskedasticity of residuals
  — White's test, etc.
- Test for autocorrelation of the residuals
  — Durbin-Watson's test, etc.
- Test for normality of the residuals
  — any goodness-of-fit tests
- Analysis if out-of-sample mean squared error (mean squared prediction error) and out-of-sample residuals