Introduction to Machine Learning

Valentina Sessa

Nov 2022
CMA - Sophia Antipolis

Machine Learning

what society thinks I do

what my friends think I do

what my parents think I do

$$L_p = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{w} + b) + \sum_{i=1}^{l} \alpha_i$$

$$\alpha_i \geq 0, \forall i$$

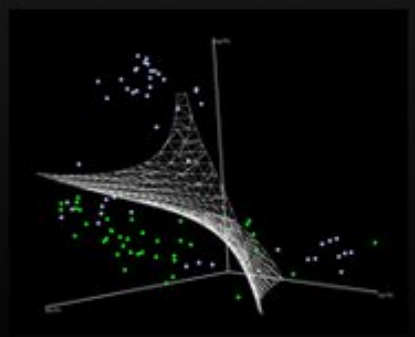
$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i, \sum_{i=1}^{l} \alpha_i y_i = 0$$

$$\nabla \hat{g}(\theta_t) = \frac{1}{n} \sum_{i=1}^{n} \nabla \ell(x_i, y_i; \theta_t) + \nabla r(\theta_t).$$

$$\theta_{t+1} = \theta_t - \eta_t \nabla \ell(x_{i(t)}, y_{i(t)}; \theta_t) - \eta_t \cdot \nabla r(\theta_t)$$

$$\mathbb{E}_{i(t)}[\ell(x_{i(t)}, y_{i(t)}; \theta_t)] = \frac{1}{n} \sum_i \ell(x_i, y_i; \theta_t).$$

what other programmers think I do



what I think I do

```
>>> from scipy import SVM
```

what I really do

Introduction

We can define **machine learning** as:

- a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty.
- a computer program conceived to learn some class of tasks T from experience E with respect to performance measure P (Mitchell, 1997).
- a form of applied statistics with increased emphasis on the use of computers to statistically estimate complicated functions and a decreased emphasis on proving confidence intervals around these functions (Godfellow, 2016).

Typical tasks

- Classification: specifying which of k categories some input belongs to.
- Regression: predicting a numerical value given some input.
- Machine translation: given a sequence of symbols in some language, the algorithm must convert it into a sequence of symbols in another language.
- Anomaly detection: The algorithm sifts through a set of events or objects and flags some of them as being unusual or atypical.
- Denoising: given a corrupted example obtained by an unknown corruption process, the algorithm must provide a clean version of this example.

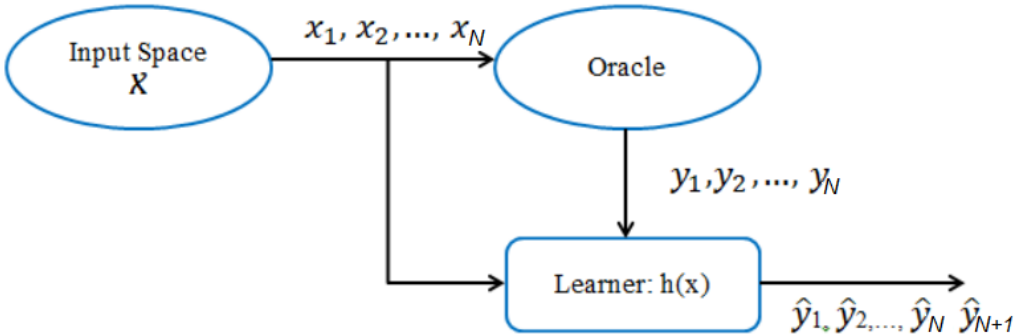
Learning based on experience

- Supervised Learning: the learning process is guided by the teacher's information (the knowledge of some output value guides the learning process)
  - Classification: the output values are categorical.
  - Regression: the output values are quantitative.
- Unsupervised Learning: these algorithms discover hidden patterns or data groupings without the need for external intervention. (ex. clustering)
- Reinforcement learning: learning how to act or behave when given occasional reward or punishment signals.

Supervised learning

Let us consider:

- the input values: $x_1, x_2, ..., x_d$, where $x_j \in \mathcal{X}_j$ corresponds to the value of the input variable X_j. $\mathbf{x} := (x_1, x_2, ..., x_d)$ forms a d-dimensional input vector x_i taking its values in $\mathcal{X}_1 \times \mathcal{X}_2 ... \times \mathcal{X}_d = \mathcal{X}$, where $\mathcal{X}$ defines the input space.
- the output value: $y_i \in \mathcal{Y}$ is the value of the output variable Y, where $\mathcal{Y}$ is the output space.
- the training set: $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$, where the pairs $(\mathbf{x}_i, y_i)$ are drawn i.i.d. from $(\mathcal{X}, \mathcal{Y})$ according to some unknown joint distribution $P(\mathcal{X}, \mathcal{Y})$.



Let us define $\hat{y} := h(\mathbf{x})$.

What is inside the oracle?

$$y = f(x) + \epsilon$$

where ϵ is a random term, which is independent of x and has mean zero.

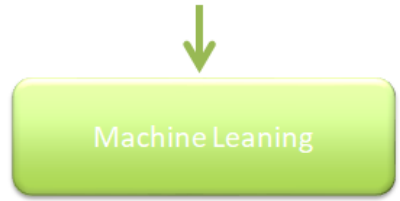
The quantity ϵ contains unmeasured variables or an unmeasurable variation. The aim of the supervised learning is to estimate f as good as possible in order to minimize the so-called *reducible error*.

However, even if it were possible to form a perfect estimate for f, our prediction will always have some error in it. This is because y also depends on ϵ, which, by definition, cannot be predicted using x.

Regression

In this case, $y \in \mathbb{R}$. Ex. Predicting hydropower generation.

HRoR	Predictor	Spatial scale	Source (Training)	Source (Forecast)
X	**Temperature** (it also includes lagged time series)	NUTS2	C2P	C2P
X	**Precipitation** (it also includes lagged time series and accumulation)	NUTS2	C2P	C2P

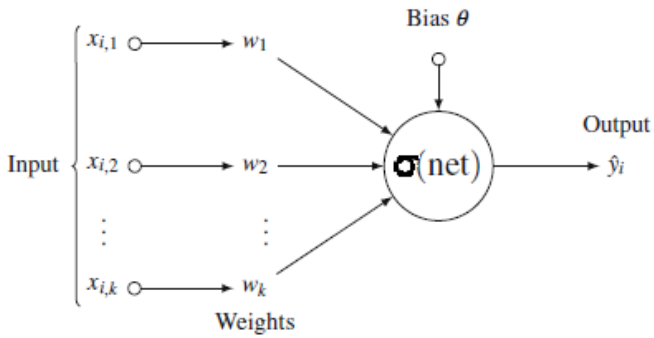


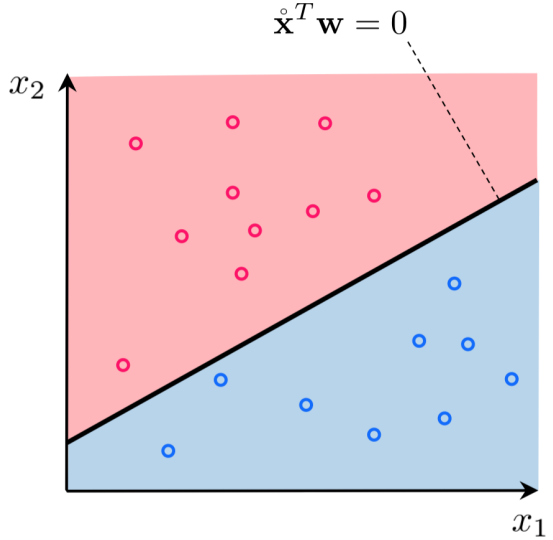
Response	Spatial scale	
HRoR	NUTS0	Seasonal forecasts
HRes		Long-term projections

- The training set $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ contains:
  - $\mathbf{x}_t = (\text{Temperature}_t, \text{Precipitation}_t)$
  - $y_t = \text{Hydropower CF}$

Classification

Perceptron (Rosenblatt, 1957)
In this case, $y \in \{-1, 1\}$, only two classes.





Classification

In this case, $y \in \{1, 2, ..., C\}$, with C being the number of classes.

Ex. Imagine you want to classify three different kinds of iris flowers based on four features: sepal length, sepal width, petal length and petal width. You get a picture of a flower, which is not in your library (the training set), what can you say?
We could compute our best guess by considering the most probable class label, that is

$$\hat{y} = h(\mathbf{x}) = \text{argmax}_{c=1}^{C} p(y = c|\mathbf{x}, S)$$

where $p(y = c|\mathbf{x}, S)$ denote the probability distribution over possible labels, given the input vector $\mathbf{x}$ and the training set S.

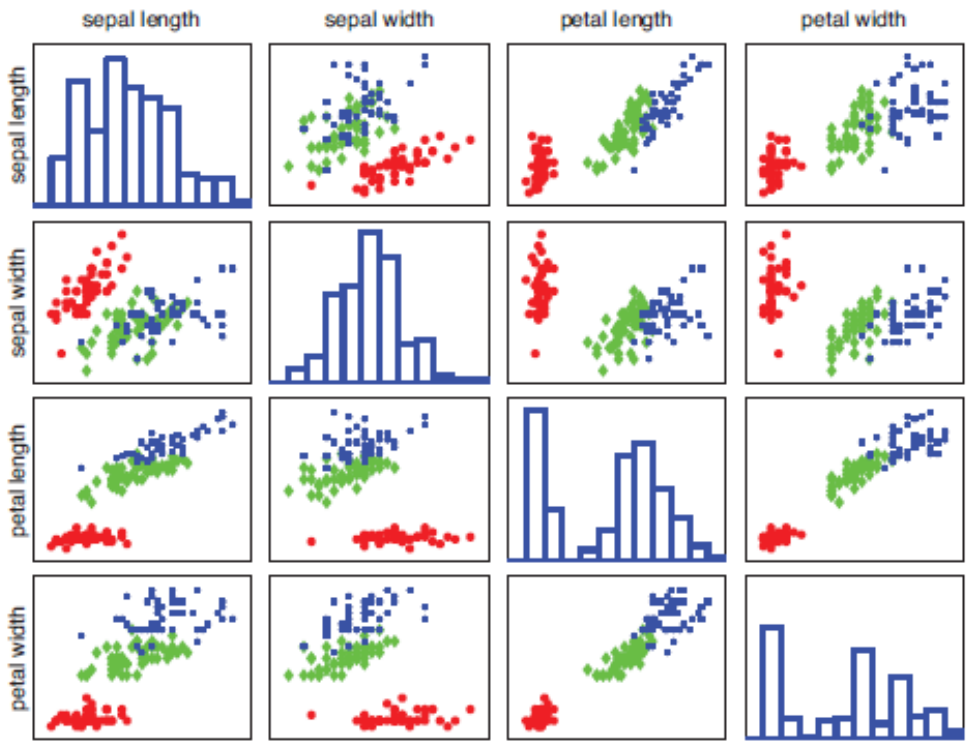


Figure: Visualization of the Iris data as a pairwise scatter plot. The diagonal plots the marginal histograms of the 4 features. The off diagonals contain scatterplots of all possible pairs of features. Red circle = setosa, green diamond = versicolor, blue star = virginica.

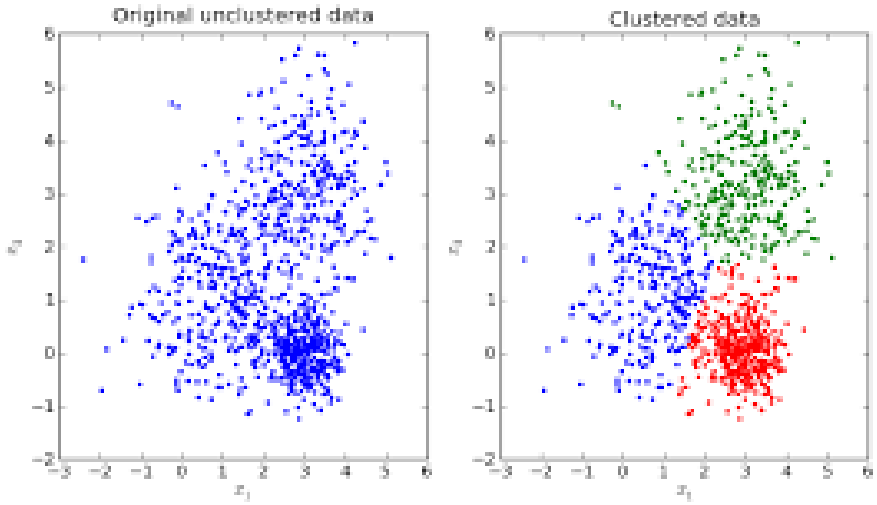
1

[1] Figure from K. Murphy, Machine Learning

Knowledge discovery

The goal of unsupervised learning is to discover 'interesting structure' in the data.

Ex. Clustering data.



Unsupervised learning

Let us consider:

- the input values: $x_1, x_2, ..., x_d$, where $x_j \in \mathcal{X}_j$ corresponds to the value of the input variable X_j. $\mathbf{x} := (x_1, x_2, ..., x_d)$ forms a d-dimensional input vector x_i taking its values in $\mathcal{X}_1 \times \mathcal{X}_2 ... \times \mathcal{X}_d = \mathcal{X}$, where $\mathcal{X}$ defines the input space.
- the training set: $S = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_d\}$, where the pairs $\mathbf{x}_i$ are drawn i.i.d. from $P_{\mathcal{X}}$.

The problem can be formalized as

$$p(\mathbf{x}_i)$$

where $p(\mathbf{x}_i)$ is an unconditional and multivariate density function.

Type of learning

- Parametric methods: Make an assumption about the shape of the function that you model a certain problem. Estimate the (fixed number of) parameters of the model. (ex. Least square)
  Good: it is mathematically tractable. Faster to use.
  Bad: The chosen shape could be very different from the true one.

- Non-Parametric methods: Do not make explicit assumptions about the functional form of f. The number of parameters grows with the amount of training data.(ex. KNN, DT).
  Good: Since there is no assumption on the shape of the f, the accuracy can improve a lot. More flexibility
  Bad: Need a lot of data! ($N \to \infty$) often computationally intractable for high dimensional inputs ($d >> 1$).

A simple non-parametric classifier: K-nearest neighbors

- "Look at" the K points in the training set that are nearest to the test input x
- Count how many members of each class are in this set
- Return that empirical fraction as the estimate

$$p(y = c|\mathbf{x}, S, K) = \frac{1}{K} \sum_{i \in N_K(\mathbf{x}, S)} \mathbb{I}(y_i = c)$$

where $N_K(\mathbf{x}, S)$ are the indices of the K nearest points to $\mathbf{x}$ in S and $\mathbb{I}(\cdot)$ is the indicator function.
How to define the distance? e.g., Euclidean distance.

KNN: example

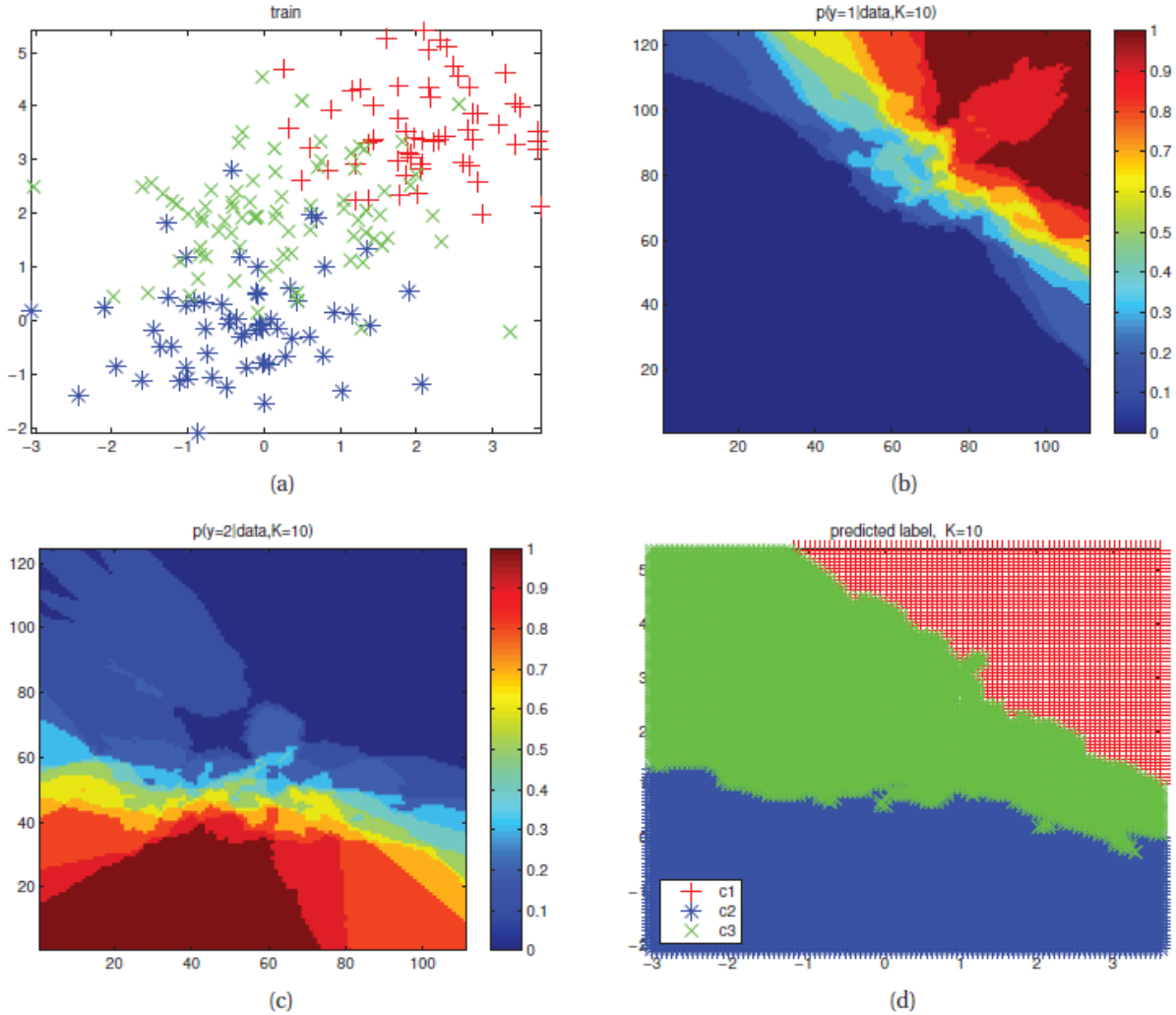


Figure: (a) Some synthetic 3-class training data in 2d. (b) Probability of class 1 for KNN with $K = 10$. (c) Probability of class 2. (d) MAP estimate of class label.

Curse of dimensionality

- The KNN classifier is simple and can work quite well, provided it is given a good distance metric and has enough labelled training data.
- The main problem with KNN classifiers as well as many machine learning methods is that they do not work well with high dimensional inputs.
- As the number of relevant dimensions of the data increases, the number of configurations of interest may grow exponentially.
- Machine leaning problems become exceedingly difficult when the number of dimensions in the data is high. This phenomenon is know as the curse of dimensionality.

Curse of dimensionality

Ex. Consider the KNN procedure for inputs uniformly distributed in a p-dimensional unit hypercube.
Consider a hypercubical neighbourhood about a target point to capture a fraction r of the observations.
Since this corresponds to a fraction r of the unit volume, the expected edge length will be $\ell = r^{\frac{1}{p}}$.

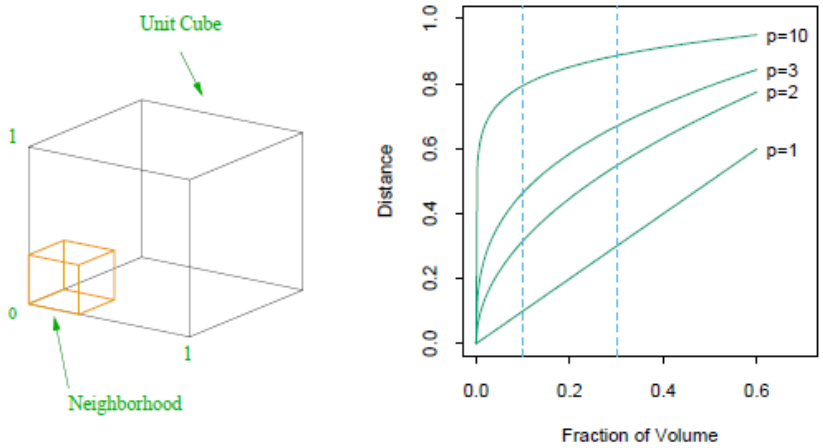


Figure: In ten dimensions we need to cover 80% of the range of each coordinate to capture 10% of the data.

Such neighbourhoods are no longer 'local' and they provide not necessarily good information.

Linear Regression

Assumption: Given $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$, we assume that the response is a linear function of the inputs, i.e.,

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + \epsilon = \sum_{j=1}^{N} w_j x_j + \epsilon,$$

where

w : weight vector

x : input vector (features)

ϵ : residual error

Linear Regression

Assumption: Given $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$, we assume that the response is a linear function of the inputs, i.e.,

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + \epsilon = \sum_{j=1}^{N} w_j x_j + \epsilon,$$

where

w : weight vector

x : input vector (features)

ϵ : residual error

We often assume that $\epsilon \sim \mathcal{N}(0, \sigma^2)$, then we can write the model describing the probability distribution of y in the following form:

$$p(y|\mathbf{x}, \theta) = \mathcal{N}(y|\mu(\mathbf{x}), \sigma^2(\mathbf{x}))$$

Simple case: $\mu(x) = \mathbf{w}^T\mathbf{x}$ and $\sigma^2(x) = \sigma^2$. So, $\theta = (\mathbf{w}^T, \sigma^2)$.

Logistic regression (classification!)

Assumption: Given $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$, let us assume that $y_i \in \{0, 1\}$ and that y_i are generated by a Bernoulli distribution, that is

$$p(y|\mathbf{x}, \mathbf{w}) = \texttt{Ber}(y|\mu(\mathbf{x}))$$

where $\mu(\mathbf{x}) = \mathbb{E}[y|\mathbf{x}] = p(y = 1|\mathbf{x})$.
In the case of logistic regression, we assume that the $\mu(\mathbf{x}) = \text{sigm}(\mathbf{w}^T\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^T\mathbf{x})}$.
Ex. Logistic regression for SAT scores

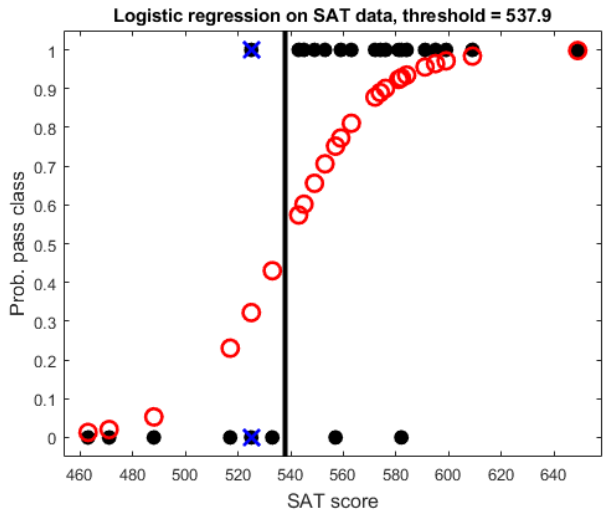


Figure: Figure generated by logregSATdemo (Machine learning : a probabilistic perspective / Kevin P. Murphy).

Performance measure

- To evaluate the performance of a machine learning algorithm, we must design a quantitative measure of its performance.
- Usually we are interested in how well the algorithm performs on data that it has not seen before, since this determines how well it will work when deployed in the real world.

Error definitions

In the statistical sense, input and output variables $X_1, ..., X_n$ and Y are random variables taking jointly their values from $\mathcal{X} \times \mathcal{Y}$ with respect to the joint probability distribution $D(X, Y)$, where X denotes the random vector $(X_1, ..., X_n)$.

Expected Prediction Error

Given the model h_S built on the training set S, we define

$$EPE(h_S) = \mathbf{E}_{(\mathbf{x},y)\sim D}[\ell(y, h_S(\mathbf{x}))]$$

$EPE(h_S)$ is the error committed by considering the model $h_S(\mathbf{x})$ over the all possible test points $(\mathbf{x}, y) \sim D$ and $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a *loss function*.

We want to estimate h as: $h^* = \text{argmin}_{h\in\mathcal{H}} EPE(h)$

Error definitions

- For classification, the most common loss function is the zero-one loss function $\ell(y, h_S(\mathbf{x})) = 1(y \neq h_S(\mathbf{x}))$, where all misclassification are equally penalized. In this case, the generalization error is the probability of misclassification:

$$EPE(h_S) = \mathbf{E}_{(\mathbf{x},y)\sim D}[\ell(y, h_S(\mathbf{x}))] = P(y \neq h_S(\mathbf{x}))$$

- For regression, the most used loss function is the squared error loss $\ell(Y, h_S(X)) = (Y - h_S(X))^2$, where large differences between the true values and the predicted values are penalized more heavily than small ones. With this loss, the generalization error of the model becomes:

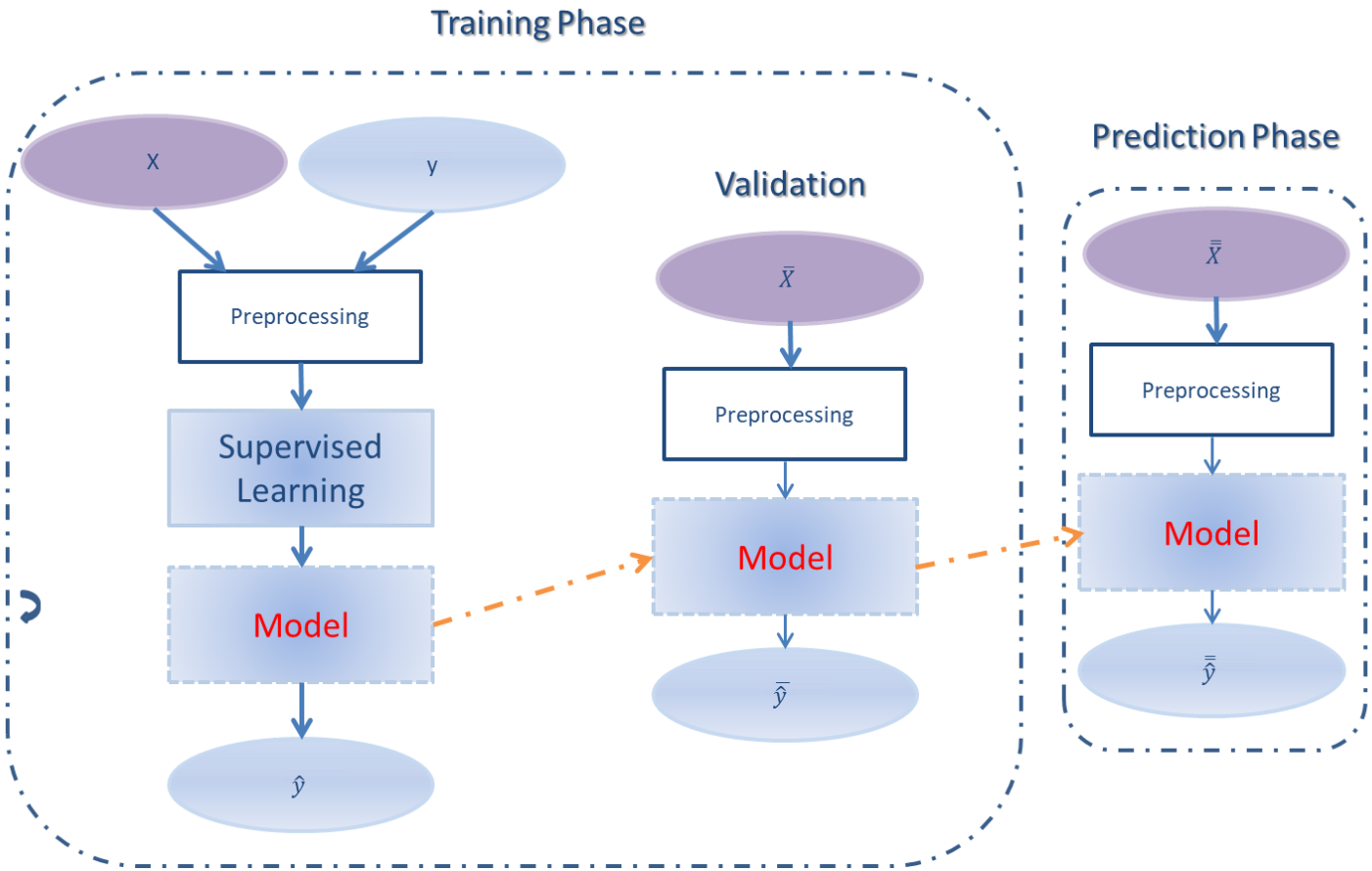
$$EPE(h_S) = \mathbf{E}_{(\mathbf{x},y)\sim D}[(y - h_S(\mathbf{x}))^2]$$

Since, in practice, the joint probability distribution $D(X, Y)$ is usually unknown, we cannot compute the 'exactly' EPE. For this reason, approximations are used. For instance, we can consider the *Average Prediction Error* defined as:

$$\overline{EPE} = \frac{1}{\bar{m}} \sum_{(x_i,y_i)\in\bar{S}} \ell(y_i, h_S(x_i)),$$

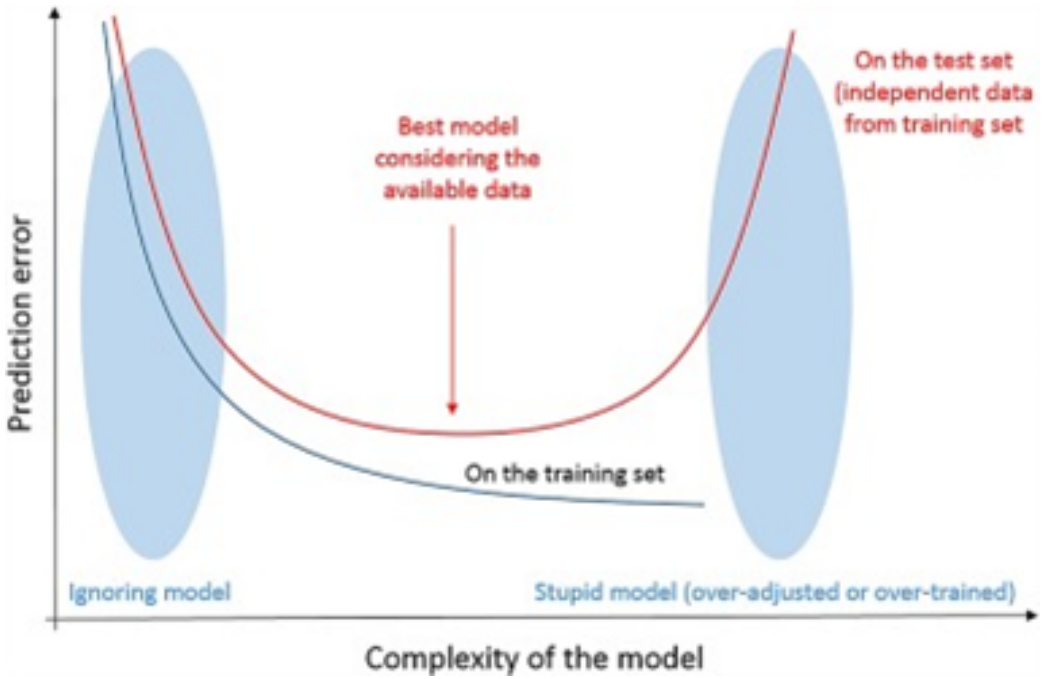
where $\bar{S}$ is a set possibly different from S.

ML workflow


Training Phase
X
y
Preprocessing
Supervised Learning
Model
$\hat{y}$
Validation
$\bar{X}$
Preprocessing
Model
$\bar{\hat{y}}$
Prediction Phase
$\bar{\bar{X}}$
Preprocessing
Model
$\bar{\bar{\hat{y}}}$

Error

The generalization performance of a learning method relates to its prediction capability on independent test data. Assessment of this performance is extremely important in practice, since it guides the choice of learning method or model, and gives us a measure of the quality of the ultimately chosen model.



Bias-variance decomposition for the generalization error

Let us consider

- $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)\}$, a training sample set drawn i.i.d. from $(\mathcal{X}, \mathcal{Y})$ according to some unknown joint distribution D.
- $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$.
- h_S the model given by an algorithm $\mathcal{A}$ considering the training set S.

Expected Label (given $\mathbf{x} \in \mathbb{R}^d$)

$$\bar{y}(\mathbf{x}) = \mathbf{E}_{y|\mathbf{x}}[Y]$$

Expected Classifier (over infinite training sets, given $\mathcal{A}$)

$$\bar{h}(\mathbf{x}) = \mathbf{E}_{S \sim D^N}[h_S(\mathbf{x})]$$

Note that S is a collection of random variables, and it is therefore a random variable. Further, h_S is a function of S, and is therefore also a random variable.

Generalization error

Generalization error (given $\mathcal{A}$)
It refers to the average test error that we would obtain if we repeatedly estimated h_S using a large number of training sets and test on a different dataset

$$\mathbf{E}_{(\mathbf{x},y)\sim D, S\sim D^N}[(y - h_S(\mathbf{x}))^2] = \int_S \int_{\mathbf{x}} \int_y (y - h_S(\mathbf{x}))^2 D(\mathbf{x}, y) D(S) \partial\mathbf{x} \partial y \partial S$$

Let's decompose it!

$$\begin{aligned}
\mathbf{E}_{\mathbf{x},y,S}[(y - h_S(\mathbf{x}))^2] &= \mathbf{E}_{\mathbf{x},y,S}[((y - \bar{h}(\mathbf{x})) + (\bar{h}(\mathbf{x}) - h_S(\mathbf{x}))^2] \\
&= \mathbf{E}_{\mathbf{x},y,S}[(\bar{h}(\mathbf{x}) - h_S(\mathbf{x}))^2] + \\
&\quad + \mathbf{E}_{\mathbf{x},y,S}[(y - \bar{h}(\mathbf{x}))^2] + 2 \underbrace{\mathbf{E}_{\mathbf{x},y,S}[(\bar{h}(\mathbf{x}) - h_S(\mathbf{x}))(y - \bar{h}(\mathbf{x}))]}_{=0}
\end{aligned}$$

$$\mathbf{E}_{\mathbf{x},y,S}[(\bar{h}(\mathbf{x}) - h_S(\mathbf{x}))(y - \bar{h}(\mathbf{x}))] = \mathbf{E}_{\mathbf{x},y}\Big[\mathbf{E}_S[(\bar{h}(\mathbf{x}) - h_S(\mathbf{x}))](y - \bar{h}(\mathbf{x}))\Big]$$

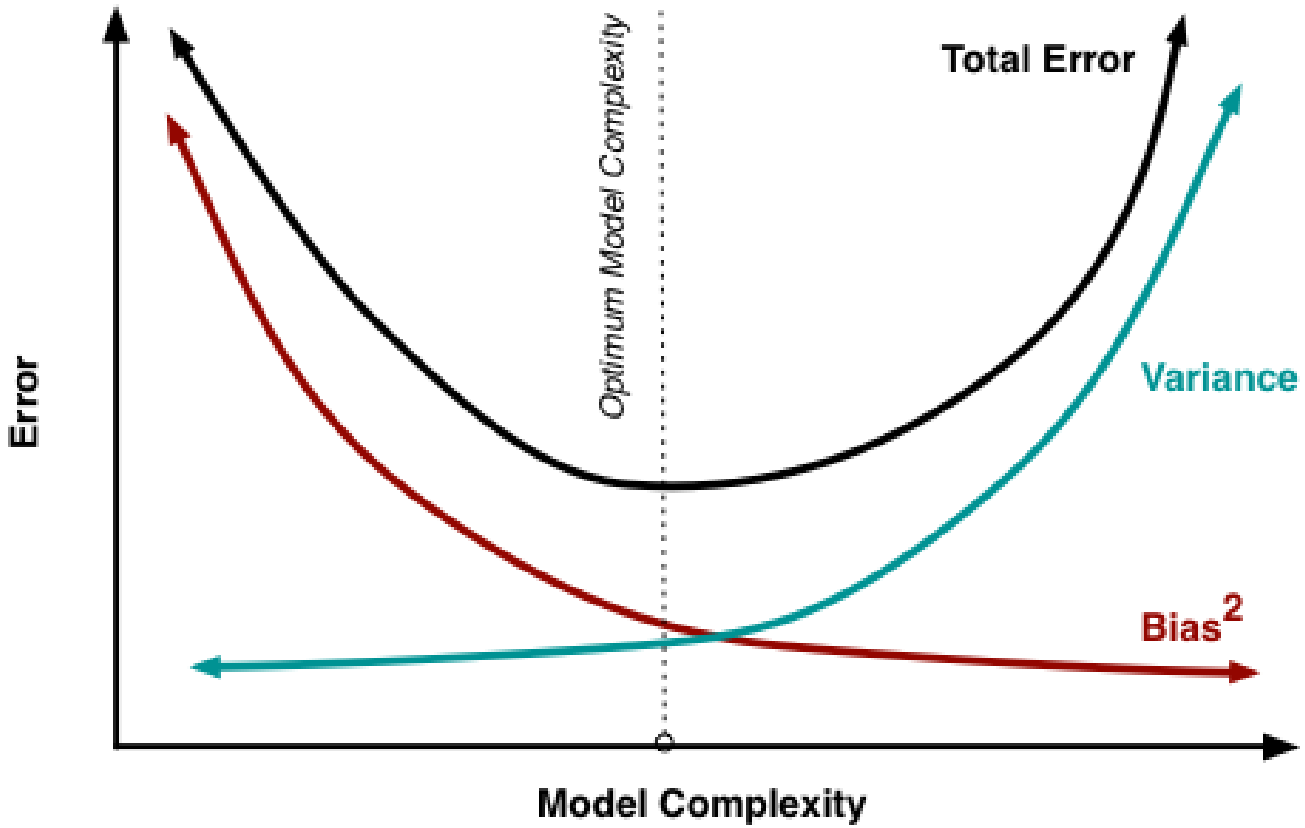
$$\mathbf{E}_S[(\bar{h}(\mathbf{x}) - h_S(\mathbf{x}))] = \mathbf{E}_S[\bar{h}(\mathbf{x})] - \mathbf{E}_S[h_S(\mathbf{x})] = \bar{h}(\mathbf{x}) - \bar{h}(\mathbf{x}) = 0$$

$$\begin{aligned}\mathbf{E}_{\mathbf{x},y}[(y-\bar{h}(\mathbf{x}))^2] &= \mathbf{E}_{\mathbf{x},y}[((y-\bar{y}(\mathbf{x}))+(\bar{y}(\mathbf{x})-\bar{h}(\mathbf{x})))^2] \\ &= \mathbf{E}_{\mathbf{x},y}[(y-\bar{y}(\mathbf{x}))^2]+\mathbf{E}_{\mathbf{x},y}[(\bar{y}(\mathbf{x})-\bar{h}(\mathbf{x}))^2]+ \\ &\quad +2\underbrace{\mathbf{E}_{\mathbf{x},y}[(y-\bar{y}(\mathbf{x}))(\bar{y}(\mathbf{x})-\bar{h}(\mathbf{x}))]}_{=0}\end{aligned}$$

So,

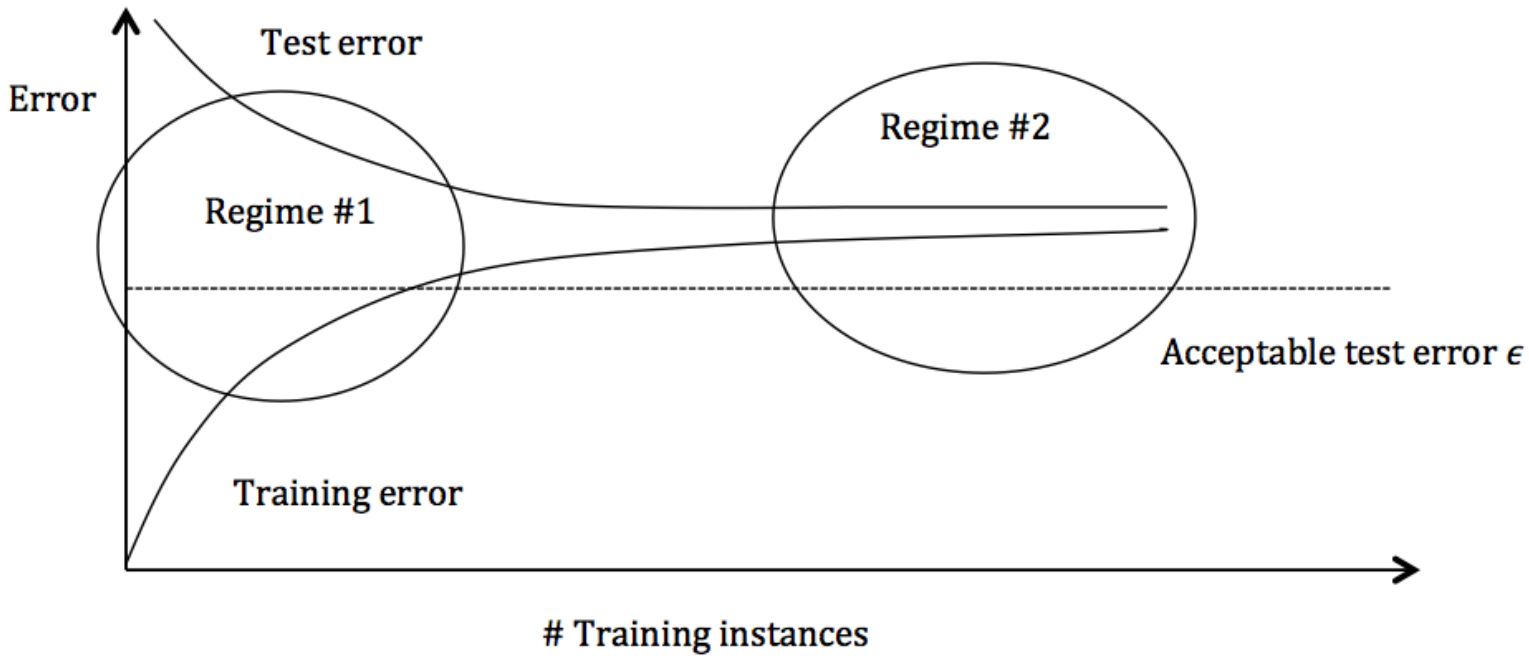
$$\mathbf{E}_{\mathbf{x},y,S}[(y-h_S(\mathbf{x}))^2] = \underbrace{\mathbf{E}_{\mathbf{x},S}[(\bar{h}(\mathbf{x})-h_S(\mathbf{x}))^2]}_{\text{variance}}+\underbrace{\mathbf{E}_{\mathbf{x}}[(\bar{y}(\mathbf{x})-\bar{h}(\mathbf{x}))^2]}_{\text{bias}^2}+\underbrace{\mathbf{E}_{\mathbf{x},y}[(y-\bar{y}(\mathbf{x}))^2]}_{noise}$$

- *Variance*: measures how much the classifier changes if we train on a different training set. It is the impact of using different datasets. How 'over-specialized' is the classifier to a particular training set (overfitting)?
- *Bias*^{2}: represents the extent to which the average prediction over all data sets differs from the expected label. It refers to the error that is introduced by approximating a real-life problem (which may be extremely complicated) by using a much simpler model.
- *Noise*: is the intrinsic error in the data, the lowest bound of generalization error for the current task that any model will not be able to get rid of, indicating the difficulty of this task.


Error
Optimum Model Complexity
Total Error
Variance
Bias^2
Model Complexity

Detecting High Bias and High Variance

If a classifier is under-performing (e.g. if the test or training error is too high), there are several ways to improve performance. To find out which of these many techniques is the right one for the situation, the first step is to determine the root of the problem.



Source:www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote12.html

Detecting High Bias and High Variance

Regime 1 (High Variance)

Symptoms:

Training error is much lower than test error

Training error is lower than ϵ

Test error is above ϵ

Remedies:

Add more training data

Reduce model complexity

Bagging

Regime 2 (High Bias)

Symptoms:

Training error is higher than ϵ

Remedies:

Increase model complexity

Add features

Boosting

Resampling methods

Estimate the variability of a model, by checking its fit on new sample drawn from the training set.

Resampling methods are used for

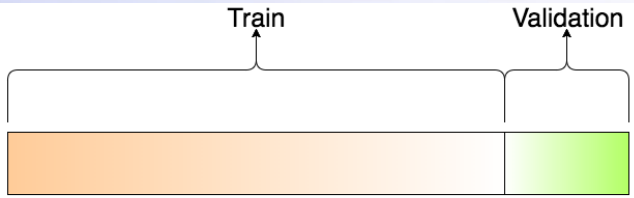
- *Model assessment*: evaluating a model's performance
- *Model selection*: selecting the level of flexibility for a model

Methods

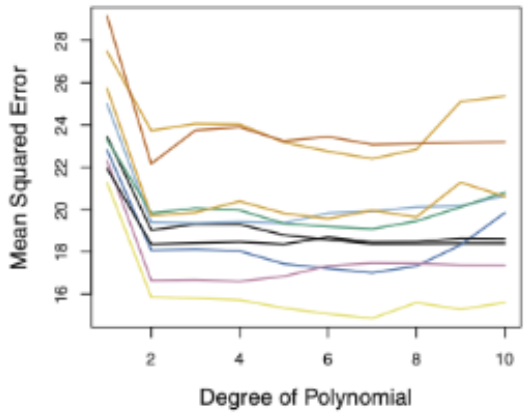
- Cross-validation
- Bootstrap

Basic idea: validation-Set

Randomly split the sample set into two sets of observations: a training set and a validation set.



If we repeat the process of randomly splitting the sample set into two parts, then we will get different estimates of the test error.



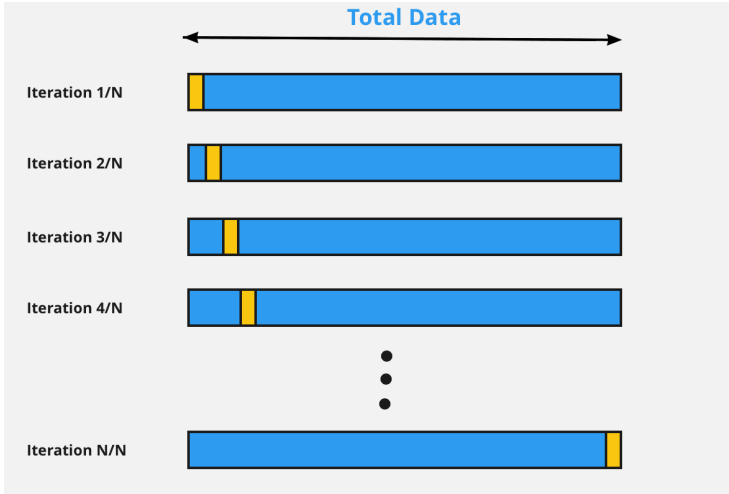
Drawbacks:

- The test error on different validation set may have high variability and a conclusion can be difficult to be given.
- If the number of training cases is small, this technique runs into problems, because the model won't have enough data to train on, and we won't have enough data to make a reliable estimate of the future performance.

Leave-one-out Cross-validation

Given $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$

- Use $N - 1$ observations for training a model
- Fit the model over the left-out observation
- Compute the error over the left-out observation
- Repeat these steps N times
- Compute the CV error as the mean of the N errors



$$MSE^{LOOCV} = \frac{1}{N}\sum_{i=1}^{N} MSE_i$$

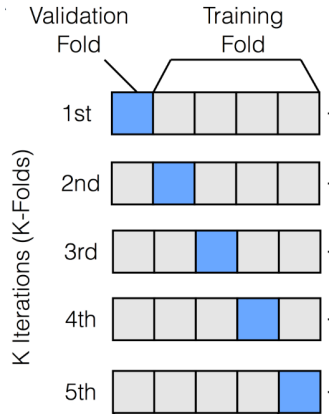
Advantages:

- Smaller bias than Validation set as we train over more data points
- The final CV error is not random

Drawback: It can be slow, as we have to train N times.

K-fold cross-validation

- Randomly divide the sample set into k folds of equal size.
- The first fold is used for as validation set, whereas $k-1$ folds are used for training a model.
- Repeat these steps k times
- Compute the k-fold CV error as the mean of the k errors.



$$MSE^{kCV} = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

It is faster the LOOCV, as the training is repeated $k < N$ times.
It may have a bigger bias, as we train over a smaller number of samples.
It may have a smaller variance than LOOCV as the training set as less correlated (there is a smaller intersection among the training sets).

Bootstrap

- In frequentist statistics, a parameter $\hat{\theta}$ is computed by applying an estimator f to some data S, so $\hat{\theta} = f(S)$. The parameters are viewed as fixed and the data as random.
- To assess the uncertainty in the parameter estimate, one can compute *the sampling distribution of the estimator*.
  - Imagine creating many different data sets $S^{(k)}$ from some true model, $p(\cdot|\theta^*)$, i.e., let $S^{(k)} = \{\mathbf{x}_j^{(i)}\}_{j=1}^N$, where $\mathbf{x}_j^{(i)} \sim p(\cdot|\theta^*)$, $k = 1 : K$ and θ^* is the true parameter.
  - Apply the estimator $\hat{\theta}$ to each $S^{(k)}$ to get a set of estimates, $\{\hat{\theta}(S^{(k)})\}$.
  - As we let $K \to \infty$, the distribution induced on $\hat{\theta}(\cdot)$ is the sampling distribution of the estimator.

A simple example[3]

- Suppose that we wish to invest a fixed sum of money in two financial assets that yield returns of X and Y, respectively, where X and Y are random quantities.
- We will invest a fraction α of our money in X, and will invest the remaining $1-\alpha$ in Y.
- In practice, we solve the stochastic program $\min_\alpha \texttt{Var}(\alpha X + (1-\alpha)Y)$ s.t. $\alpha \in S$ and we get

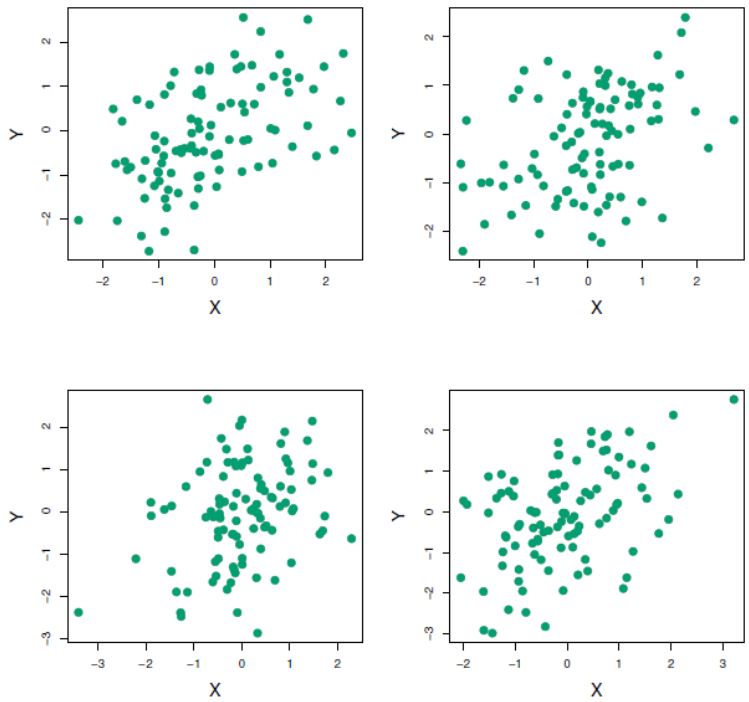
$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}},$$

  where $\sigma_X^2 = \texttt{Var}(X)$, $\sigma_Y^2 = \texttt{Var}(Y)$ and $\sigma_{XY} = \texttt{Cov}(X, Y)$, which are unknown.
- If we have a dataset that contains measurements for X and Y, then we can estimate these quantities and then we get get en estimate $\hat{\alpha}$.

[3] James, Witten, Hastie, Tibshirani An Introduction to Statistical Learning

A simple example (cont.)

We generate four sets of 100 pairs of returns for the investments X and Y and we estimate $\hat{\alpha}$ in the four cases. The resulting estimates for $\hat{\alpha}$ are 0.576, 0.532, 0.657, and 0.651.



A simple example (cont.)

- To estimate the standard deviation of $\hat{\alpha}$, we repeated the process of simulating 100 paired observations of X and Y, and estimating $\hat{\alpha}$ 1000 times.
- We thereby obtained 1000 estimates for $\hat{\alpha}$, say $\hat{\alpha}_1, \hat{\alpha}_2, ..., \hat{\alpha}_{1000}$.

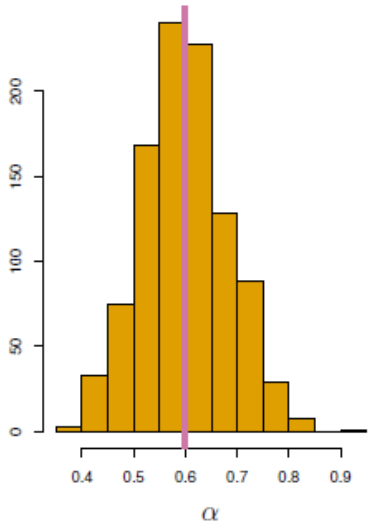


Figure: For these simulations the parameters were set to $\sigma_X^2 = 1$, $\sigma_Y^2 = 1.25$ and $\sigma_{XY} = 0.5$, and so we know that the true value of α is 0.6.

- The mean over all 1000 estimates is

$$\bar{\alpha} = \frac{1}{1000} \sum_{r=1}^{1000} \hat{\alpha}_r = 0.5996$$

and the standard deviation is

$$\sqrt{\frac{1}{1000 - 1} \sum_{r=1}^{1000} (\hat{\alpha}_r - \bar{\alpha})^2} = 0.083$$

- The obtained estimation is quite good!

Back to the real world

- The procedure outlined above cannot be applied, because for real data we cannot generate new samples from the original population.
- However, the bootstrap approach allows us to use a computer to mimic the process of obtaining new data sets, so that we can estimate the variability of our estimate without generating additional samples.
- Rather than repeatedly obtaining independent data sets from the population, we instead obtain distinct data sets by repeatedly sampling observations from the original data set with replacement.
- Each of these bootstrap data sets is created by sampling *with replacement*, and is the *same size* as our original dataset. As a result some observations may appear more than once in a given bootstrap data set and some not at all.

Creating bootstrap data sets

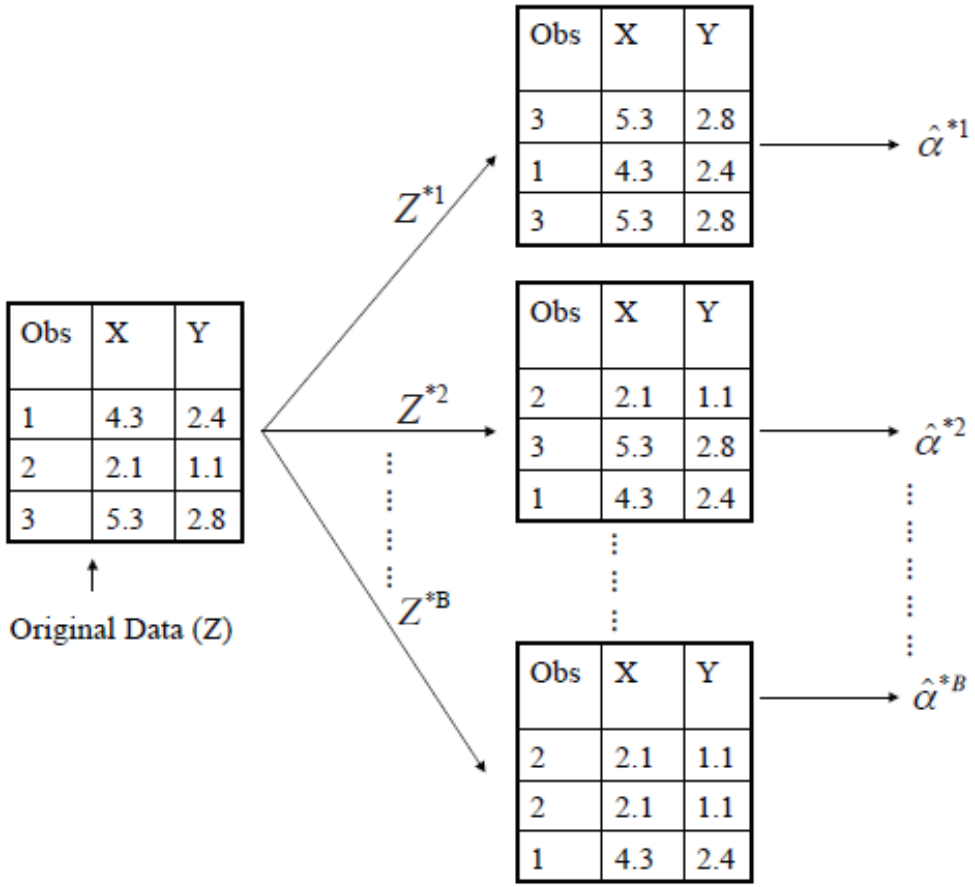


Figure: Each bootstrap data set contains n observations as the original data set and is sampled with replacement. Each bootstrap data set is used to obtain an estimate of α.

Results

The standard deviation of the estimates obtained with the bootstrap technique is 0.087, very close to the value obtained with the 1000 simulated dataset.

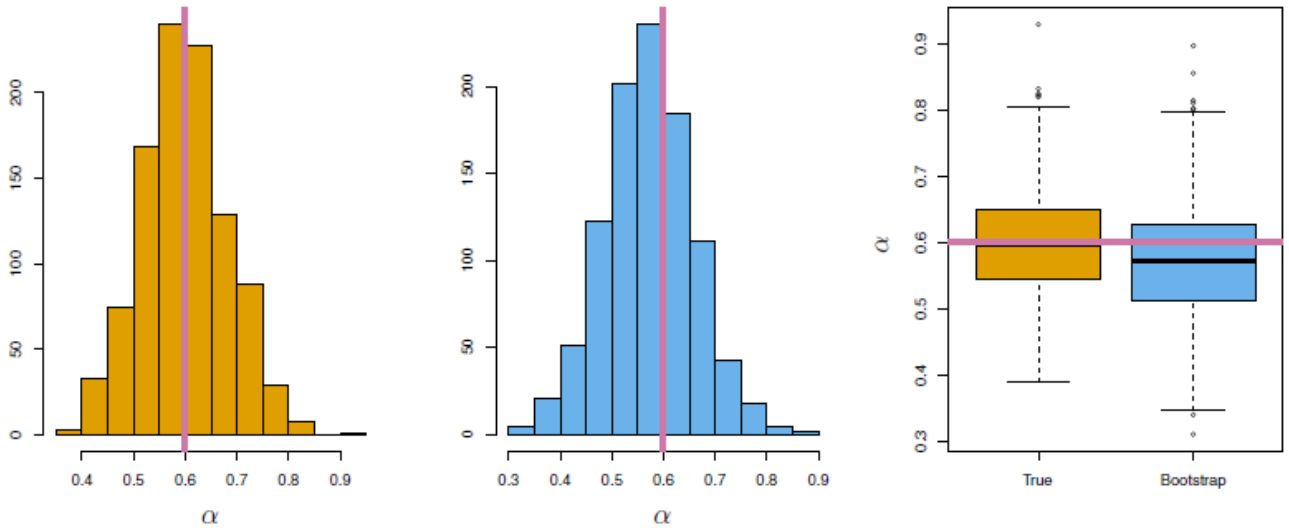
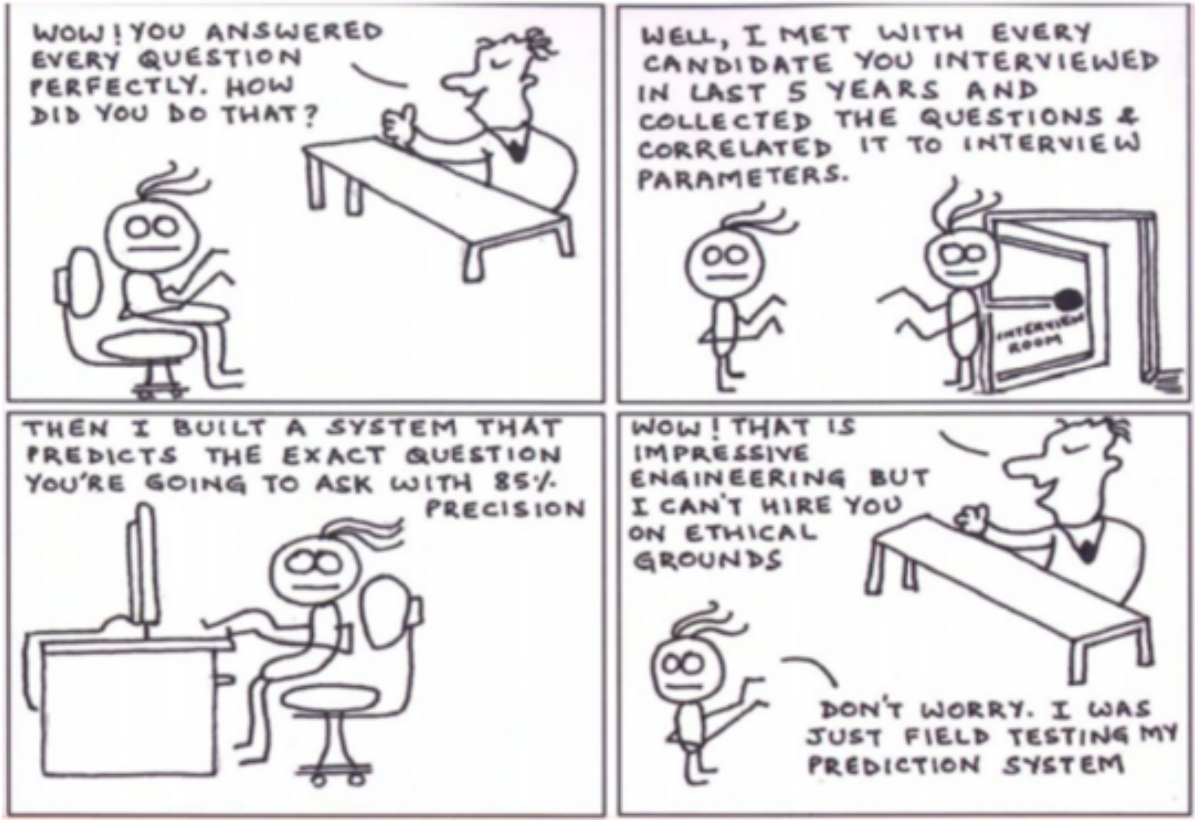


Figure: Left: A histogram of the estimates of α obtained by generating 1000 simulated data sets from the true population. Center: A histogram of the estimates of α obtained from 1000 bootstrap samples from a single data set. Right: The estimates of α displayed in the left and center panels are shown as boxplots. In each panel, the pink line indicates the true value of α.

No free lunch theorem

All models are wrong, but some are useful. George Box (1087)
We can use methods such as cross validation to empirically choose the best method for our particular problem. However, there is no universally best model!

When you interview a data scientist...



Email: valentina.sessa@mines-paristech.fr