Machine Learning: fundamentals and algorithms

Guillaume Metzler

Laboratoire Hubert Curien, UMR CNRS 5516
University of Jean Monnet Saint-Étienne (France)

my contact: guillaume.metzler@univ-st-etienne.fr

What is Machine Learning?

Born from the ambitious goal of **Artificial Intelligence**:

Can Machines Think?

Step by step, the goal has changed:

Can machines do what we (as thinking entities) can do? (A. Turing)

Tom Mitchell provided a more formal definition (1998)

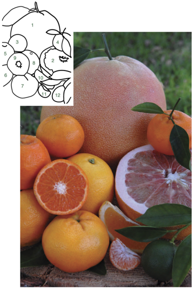
A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E

Definition

Machine Learning

Machine learning explores the construction and study of algorithms that can learn from and make predictions on data.

→ Let's take an example: Citrus Recognition



http://www.nature.com/nbt/journal/v32/n7/full/nbt.2906.html

Example: Citrus Recognition

1/ Data Represetation

pictures: matrix of pixels, channels of colors (color intensity per pixel)

$$R = \begin{bmatrix} 1 \\ 1 \\ 0.9 \\ 0 \\ .. \end{bmatrix} \quad (1) \qquad G = \begin{bmatrix} 1 \\ 1 \\ 0.1 \\ 0.3 \\ .. \end{bmatrix} \quad (2) \qquad B = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0.7 \\ .. \end{bmatrix} \quad (3)$$

Example: Citrus Recognition

2/ Hand-crafted Features

Example: Diameter, Shape, Color... but

1. diameter varies with the view distance
2. shapes are difficult to represent
3. color varies with light exposure

Another Example: Scale-Invariant Feature Transform (**SIFT**)

2/ Feature Learning

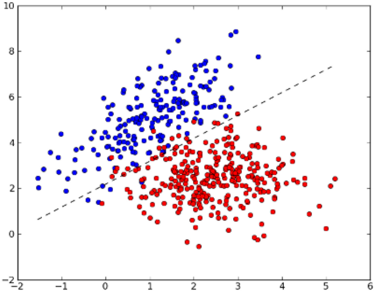
Discover useful features or representations from raw data.
Example: Neural Networks, Autoencoders, Matrix Factorization, ...

3/ Learning as Training

Input: vectors of features

Output: classes

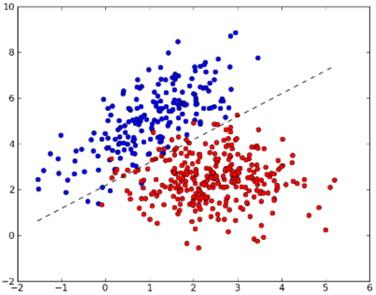
Training Data: set of input and output on which the model is learned.



training error $= 1-$ **training accuracy**

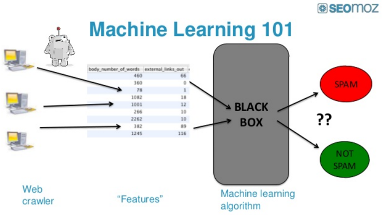
3/ Testing

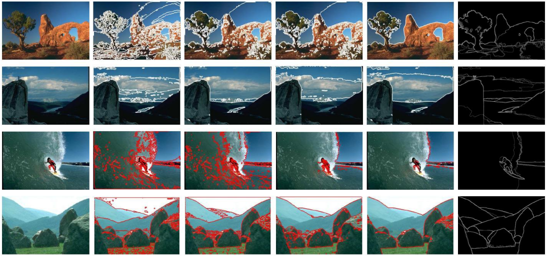
Does the model **generalize** on new (unseen) data?



test error $= 1-$ **test accuracy**

Other applications





Outline

Machine Learning Settings

Machine learning tasks are typically classified into three broad categories:

Main references

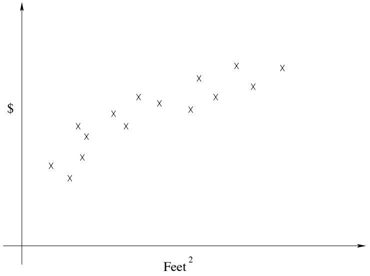
- Supervised learning
- Unsupervised learning
- Reinforcement learning

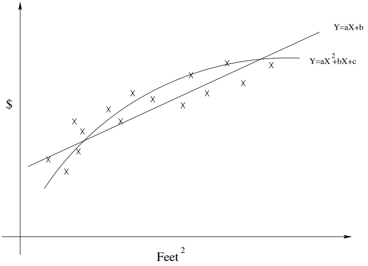
Supervised Learning: Regression

Supervised learning

The computer has access to training input examples and their desired outputs, given by a "teacher" or an "oracle". The aim is to learn a general rule that maps inputs to outputs. Once learned, the rule can be deployed on test data.

outputs = continuous values



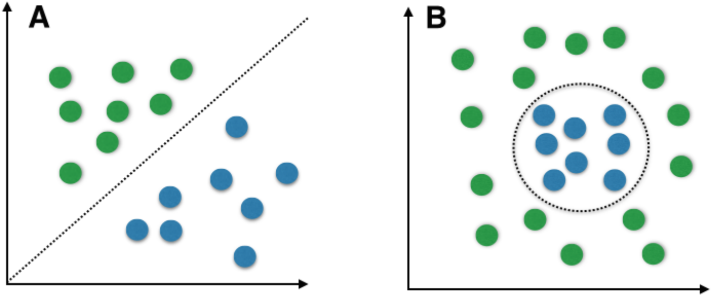


Supervised Learning: Classification

Supervised learning

The computer has access to training input examples and their desired outputs, given by a ”teacher” or an “oracle”. The aim is to learn a general rule that maps inputs to outputs. Once learned, the rule can be deployed on test data.

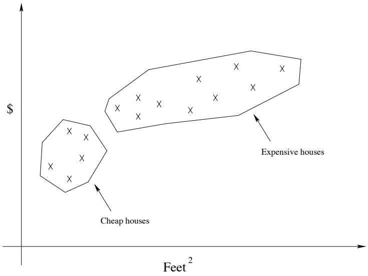
outputs = discrete values (labels)



Unsupervised Learning: Clustering

Unsupervised learning

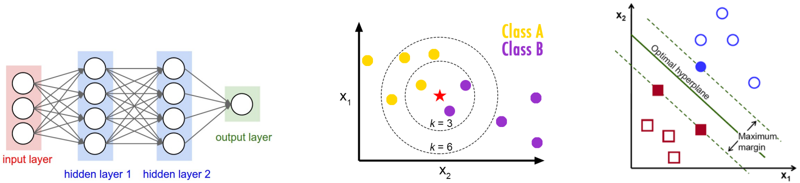
The output are not provided in the learning phase. The goal is to discover groups of similar examples within the data (**clustering**), or to determine the distribution of data within the input space (**density estimation**), or to project the data from a high-dimensional space down to two or three dimensions (**dimensionality reduction**).



Popular Supervised Learning Algorithms

d is the number of dimensions of the input space and n is the number of training instances.

1. Linear Regression (learn $\{\theta_i\}_0^d$)
2. Support Vector Machine (learn m weights)
3. Neural Networks (learn layers of input weights)
4. Decision Trees (learn the decision tree itself)
5. k-Nearest Neighbors (memorize the training data)

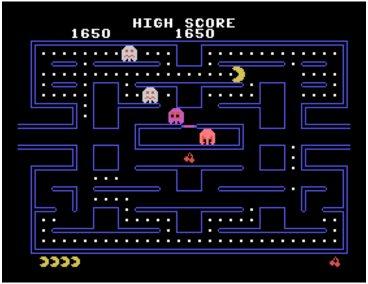


Reinforcement Learning

What are talking about ?

Reinforcement learning is concerned with the problem of finding suitable **actions** (decisions) to take in a given situation (**observations**) in order to **maximize a reward**.

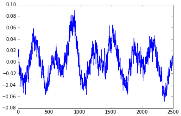
Here, the learning algorithm is not given examples of optimal outputs but must instead discover them by a process of **trial and error** (example: https://www.youtube.com/watch?v=CIF2SBVY-J0.)

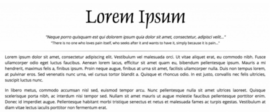


Key Ingredient in Machine Learning

Machine Learning and Data

The key ingredient of machine learning is... **DATA**, stored in many forms (and formats...), structured, unstructured, occasionally clean, usually messy,...





Lorem Ipsum

"Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit..."

"There is no one who loves pain itself, who seeks after it and wants to have it, simply because it is pain..."

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum vel malesuada orci. Ut vitae aliquet erat, eu commodo nibh. Fusce ullamcorper odio eget luctus faucibus. Aenean lorem lacus, efficitur sit amet quam eu, bibendum pellentesque ipsum. Mauris a mi hendrerit, maximus felis a, finibus ipsum. Proin neque augue, finibus at urna sit amet, facilisis ullamcorper nulla. Duis non tempus lorem, at pulvinar eros. Sed venenatis nunc urna, vel cursus tortor blandit a. Quisque et rhoncus odio. In est justo, convallis nec felis ultricies, suscipit luctus nunc.

In libero metus, commodo accumsan nisl sed, euismod tempor arcu. Nunc pellentesque nulla sit amet ultrices laoreet. Quisque scelerisque porta nibh, ac interdum nisi tempor sed. Nullam sit amet mauris ut augue molestie faucibus pellentesque porttitor enim. Donec id imperdiet augue. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum at diam vitae lectus iaculis porttitor non fermentum erat.

In ML we like to view data as a list of n examples of the same nature, preferably in the form of d-dimensional feature vectors $\boldsymbol{x} = (x_1, x_2, ..., x_d)$ with $\boldsymbol{x} \in \mathbb{R}^d$.

Data Representation

Choice of data representation is problem dependent.

pictures...but also

genetic samples as sequences of genes (dimension = gene, vectors of occurrences)

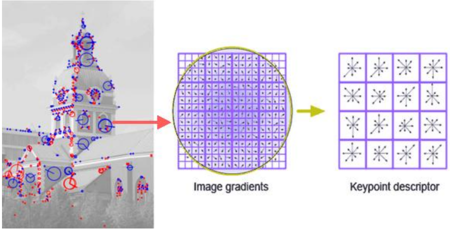
sounds as time series of signals (dimension = frequency, vectors of amplitudes)

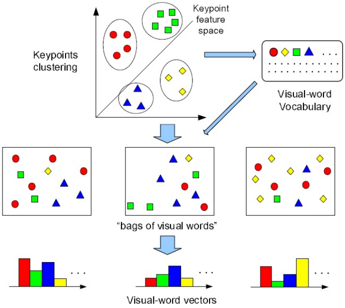
text documents as set of words (dimension = word of vocabulary, vectors of occurrences)

and **Metadata**: authors, date, ...

Real-world data is complex, redundant, and highly variable. It is necessary to discover useful features or representations from raw data.

Example: SIFT descriptors + Bag Of Words





Feature Scaling

The range of values of raw data varies widely. When computing some measures, few features can take over all the others.

Example: given two instances $\boldsymbol{x_i}$, $\boldsymbol{x_j}$ with features $x\_1 \in [0,5]$, $x\_2 \in [100,200]$, the Euclidean distance $\sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2}$ is governed by feature x_2.

Solutions

1. **scaling** (each feature, usually in the range $[-1,1]$ or $[0,1]$)
2. **standardization** (each feature, $\mu = 0$, $\sigma = 1$)
3. **normalization** (usually using l_2 norm)

Independent and Identically Distributed instances

While learning, we usually don't have access to the entire distribution of the data, but only a sample. In order to guarantee that the **model learned** on our limited sample **can generalize** on the entire distribution (and then on unseen instances), we need to assume that the **sample is i.i.d.**.

I.I.D. instances

A sample S of instances drawn from a distribution $\mathcal{D}$ is said i.i.d. if each instance has the same probability distribution as the others and all are mutually independent.

Curse of Dimensionality

Learning good generalizations is possible when $n >> d$.

Fundamental Properties of Probability

The curse of dimensionality refers to various phenomena that arise when analyzing and organizing data in **high-dimensional spaces** (often with hundreds or thousands of dimensions) that do not occur in low-dimensional settings.

$d > n$?

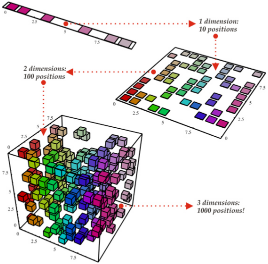
As the number of features grows, the amount of data we need to generalize accuratly grows exponentially .

Curse of Dimensionality

Corollary 1

When the dimensionality increases, the volume of the space increases so fast that the available data become sparse.

Example: $10^2 = 100$ evenly-spaced sample points suffice to sample a unit interval (a "1-dimensional cube") with no more than 0.01 distance between points; an equivalent sampling of a 10-dimensional unit hypercube with a lattice that has a spacing of 0.01 between adjacent points would require 10^{20} sample points!



Curse of Dimensionality

Corollary 2

When a measure such as a Euclidean distance is defined using many coordinates, there is little difference in the distances between different pairs of samples. This phenomenon can have a considerable impact on various techniques for classification (including the k-NN classifier) and clustering.

Example: Let us compare the proportion of an inscribed hypersphere with radius r and dimension d, to that of a hypercube with edges of length $2r$.

- The volume V_s of the hypersphere is $V_s = \frac{2r^d\pi^{(d/2)}}{d\Gamma(d/2)}$.
- The volume V_c of the hypercube is $V_c = (2r)^d$.

As the dimension d increases, the hypersphere becomes an insignificant volume relative to that of the hypercube. Indeed,

$$\lim_{d\to\infty} \frac{V_s}{V_c} = 0$$

.

Curse of Dimensionality

How to overcome it?

When facing the curse of dimensionality, a good solution can often be found by **pre-processing** the data into a lower-dimensional space. We can make use of **dimensionality reduction** methods such as Principal Component Analysis. Note that there is a huge literature about feature selection.

Supervised Learning

Notations

Let S be a set of m training examples $\{z_i = (\boldsymbol{x_i}, y_i)\}_{i=1}^m$ independently and identically (i.i.d.) from an unknown joint distribution $D_{\mathcal{Z}}$ over a space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$.

1. The x_i values ($\boldsymbol{x_i} \in X$) are typically vectors of the form $< x_{i1}, ..., x_{id} >$, whose components are usually called features.
2. The y values ($y \in Y$) are drawn from a discrete set of classes (typically $Y = \{-1, +1\}$ in binary classification) or are continuous values (regression).
3. We assume that there exists a target function f such that $y = f(\boldsymbol{x})$, $(\boldsymbol{x}, y) \in \mathcal{Z}$.

Definition

A supervised learning algorithm **L** automatically outputs from S a classifier (or a hypothesis) $h \in \mathcal{H}$ about the target function f.

True Risk and Empirical Risk

In order to pick the best hypothesis h^*, we need a criterion to assess the quality of any hypothesis h.

True Risk

The true risk $\mathcal{R}(h)$ (also called **generalization error**) of a hypothesis h corresponds to the expected error made by h over the entire distribution $D_{\mathcal{Z}}$:

$$\mathcal{R}(h) = \mathbb{E}_{z=(x,y)\sim D_{\mathcal{Z}}} \mathbb{1}_{y \neq h(x)}$$

where $z \sim D_{\mathcal{Z}}$ denotes that z is drawn i.i.d. from $D_{\mathcal{Z}}$.

The goal of supervised learning then becomes finding a hypothesis h that achieves the smallest true risk. Unfortunately, $\mathcal{R}(h)$ cannot be computed because $D_{\mathcal{Z}}$ is unknown. We can only measure it on the training sample S. This is called the **empirical risk**.

True Risk and Empirical Risk

Empirical Risk

Let $S = \{z_i = (\boldsymbol{x_i}, y_i)\}_{i=1}^m$ be a training sample. The empirical risk $\hat{\mathcal{R}}(h)$ (also called empirical error) of a hypothesis $h \in H$ corresponds to the **expected error** suffered by h on the instances in S.

$$\mathcal{R}(h) = \mathbb{E}_{\{z_i=(\boldsymbol{x_i},y_i)\}_{i=1}^m} \mathbb{1}_{y \neq h(x)}$$

where $z \sim D_{\mathcal{Z}}$ denotes that z is drawn i.i.d. from $D_{\mathcal{Z}}$.

True Risk and Empirical Risk

A **loss function** $\mathcal{L} : H \times \mathcal{Z} \to \mathbb{R}^+$ measures the degree of agreement between $h(\boldsymbol{x})$ and y.

0\1 loss or Classification Error

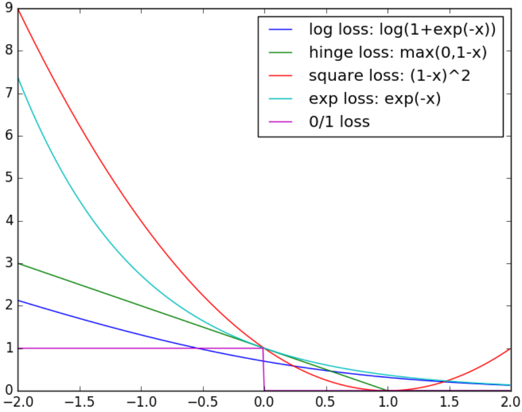
$$\mathcal{L}(h(\boldsymbol{x}), y) = \mathbb{1}_{y \neq h(\boldsymbol{x})}$$

corresponds to the proportion of time $h(\boldsymbol{x})$ and y agree, i.e. the proportion of correct predictions.

In binary classification,

$$\mathcal{L}(h(\boldsymbol{x}), y) = \begin{cases} 1 & \textit{if } h(\boldsymbol{x})y < 0 \\ 0 & \textit{otherwise} \end{cases}$$

Surrogate Losses



Empirical Surrogate Risk Minimization

Minimize the empirical risk to choose the hypothesis $h \in \mathcal{H}$:

$$h = \underset{h_i \in \mathcal{H}}{\arg\min}\, \hat{\mathcal{R}}(h_i)$$

with:

Empirical Surrogate Risk

$$\hat{\mathcal{R}}^{\mathcal{L}}(h(\boldsymbol{x}), y) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(h(\boldsymbol{x_i}), y)$$

.

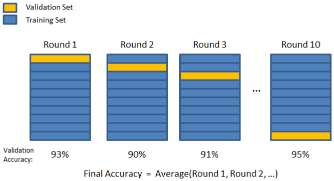
Cross-Validation

How to estimate the performances/quality of a learned model?

Cross-Validation

A way to check if the learned model generalizes well on unseen data.
Split the training set into k folds and repeat k times the following steps:

- train a model on $k-1$ folds;
- test the learn model on the 1 fold not used for training (**validation set**).



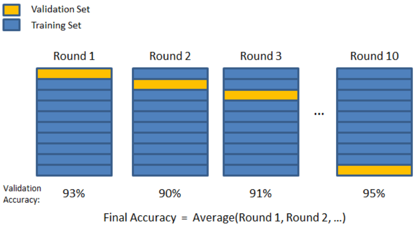
10-fold cross-validation

Cross-Validation

How to estimate the performances/quality of a learned model?

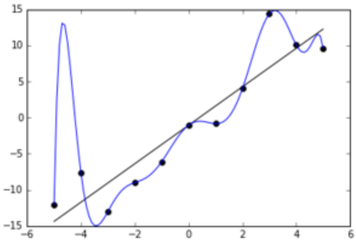
Cross-validation error

$\hat{\mathcal{R}}_{cv}(h) = \frac{1}{k}\sum_{j=1}^{k}\hat{\mathcal{R}}_j(h)$ gives an idea of how well the hypothesis h is suited for predictions on unseen data $\mathcal{S}_u$ drawn from the same distribution $\mathcal{D}_{\mathcal{Z}}$ as the available data.



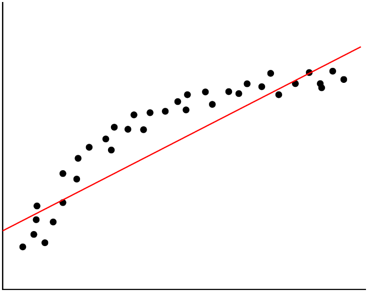
Overfitting

In statistics, overfitting occurs when a model describes random error or noise instead of the underlying relationship. Overfitting generally occurs when a **model is excessively complex** or the **size of the training dataset is small**, such as having too many degrees of freedom w.r.t. the amount of available data.

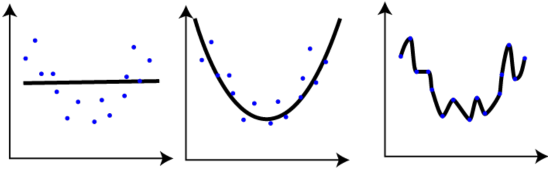


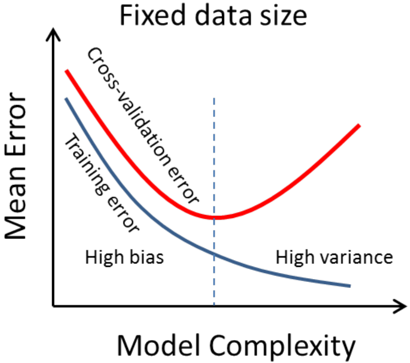
Underfitting

Underfitting occurs when a statistical model or machine learning algorithm cannot capture the underlying trend of the data. Underfitting generally occurs when a model is **excessively simple**.



Underfitting-Overfitting





How can we know if the model is underfitting or overfitting the data?

- if training error $<<$ cv error, **overfitting**
- if training error $>$ cv error or high, **underfitting**

Regularization

Occam's Razor

Choose the simplest explanation consistent with past data:

"no sunt multiplicanda entia praeter necessitatem" (William of Ockham)

(Entities are not to be multiplied beyond necessity)

W. Ockham: Born around 1285, he was an English philosopher and monk.

Regularization

A way of avoiding overfitting

Regularization

Regularization, in mathematics and statistics and particularly in the fields of machine learning, refers to a process of **introducing additional information** in order to solve an ill-posed problem or to prevent overfitting. This information is usually of the form of a **penalty for complexity**, such as restrictions for smoothness or bounds on the vector space norm.

Regularized Risk Minimization

New optimization problem:

$$h = \underset{h_i \in \mathcal{H}}{\arg\min}\, \hat{\mathcal{R}}(h_i) + \lambda ||h_i||$$

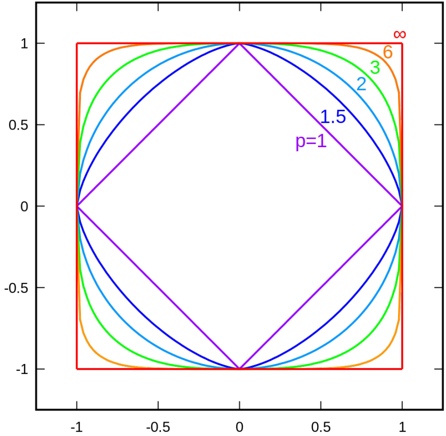
where

- λ is the regularization parameter (or hyper-parameter)
- $||.||$ is a norm function

We select a hypothesis h that achieves the best trade-off between empirical risk minimization and regularization.

Common Norms

$$||h||_p = \Big(\sum_{i=1}^{d} h_i^p\Big)^{1/p}$$

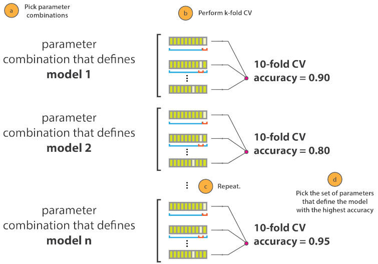


Tuning

How to choose λ?

Hyper-Parameter Tuning

1. Bad idea: choose the one with the lowest training error (problem of overfitting).
2. **Good idea: hold-out k cross-validation and select the value for hyper-parameter with the lowest cross-validation errror.**



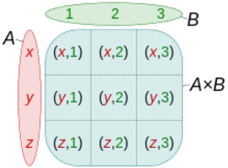
Tuning

Grid Search

A way to choose the combinations of values for multiple hyper-parameter tuning (p):

1. fix the set s_z of possible values per hyper-parameter λ_z (ex. $s_1 = \{0.001, 0.01, 0.1, 1, 10, 100\}$);
2. compute a cross-validation for each combination of values $(\lambda_1, \lambda_2, \dots)$;
3. select the combination of values $(\lambda_1, \lambda_2, \dots)$ that gives the best error.

Total number of cross-validations: $\prod_{z=1}^{p} |s_z|$.



Conclusions

Supervised Learning Process

Given a training sample S_{train} and a test sample S_{test}:

1. pre-processing of S_{train} and S_{test} separately
   1. feature scaling
   2. dimensional reduction
   3. etc...
2. learning best model on S_{train}
   1. **tuning** hyper-parameters **on training set** by cross-validation (split training/validation sets)
   2. select best values for hyper-parameters
   3. **train the algorithm on the entire training set** using the learned values for the hyper-parameters
3. get algorithm performances as the test error

Conclusions

Supervised Learning Process

Given a sample S:

1. pre-processing of S
   1. data standardization
   2. dimensional reduction
   3. etc...
2. k-fold cross-validation on S (split training/test sets)
   1. **tuning** hyper-parameters **on training set** by cross-validation (split training/validation sets)
   2. select best values for hyper-parameters
   3. **train the model on the training set** using the learned values for the hyper-parameters
   4. get **model performances on test set**
3. get algorithm performances as the cross-validation error