CENTURI
TURING CENTRE
FOR LIVING SYSTEMS

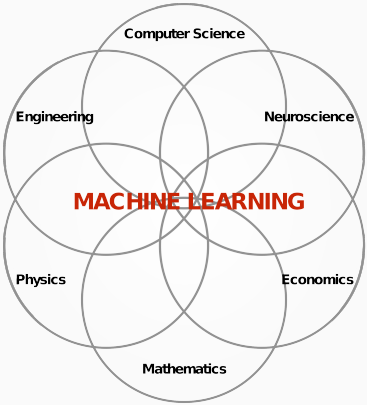
CenTuri Course 2022

Supervised Learning

Stefania Sarno

May 10, 2022

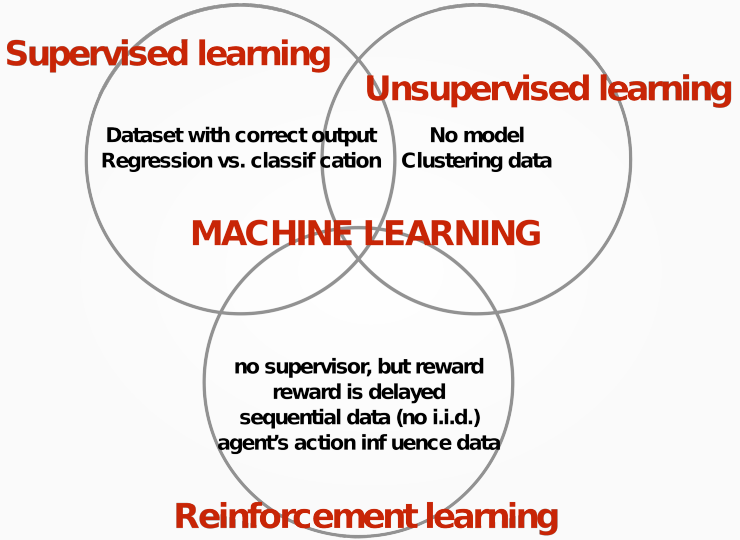
What is machine learning?



"The field of study that gives computers the ability to learn without being explicitly programmed."
-Arthur Samuel

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." -Tom Mitchell

Branches of machine learning

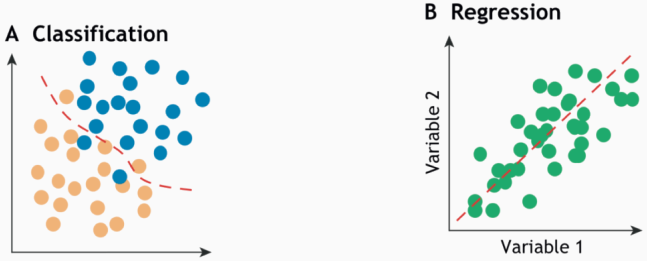


Some examples of machine learning problems

Classify emails as spam or not	Supervised Learning (Classification)
Predict the success of a movie	Supervised Learning (Regression)
Diagnose from list of symptoms	Supervised Learning (Classification)
Find groups in a social network	Unsupervised Learning
Drive a car autonomously	Reinforcement Learning
Defeat world champion of Go	Reinforcement Learning

Supervised learning

- Dataset with correct output
- Two main tasks: Regression and Classification



Regression

Univariate linear regression: example problem

- We are interested in the relationship between tree diameter (at breast height) and the dry leaf mass
- Table with 3 columns: taxon(angiosperm vs. gymnosperm), diameter, leaf mass (592 records)
- Reference: Enquist & Niklas, Science (2002). DOI: 10.1126/science.1066360

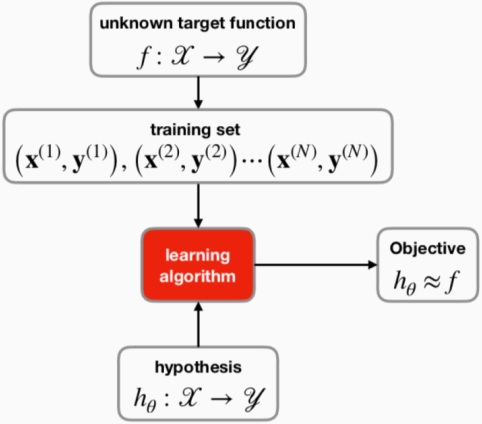
Formalization of learning

- Input: $\mathbf{x}$ (trunk diameter)
- Output: $\mathbf{y}$ (leaf mass)
- Target function: $f : \mathcal{X} \longrightarrow \mathcal{Y}$ (the relationship we are looking for)
- Data: $(\mathbf{x}^1, \mathbf{y}^1) \cdots (\mathbf{x}^N, \mathbf{y}^N)$ (to be split into training and test data)
- Hypothesis: $h_\theta : \mathcal{X} \longrightarrow \mathcal{Y}$ (the set of functions parametrized by θ)

Learning components

The two components of the **learning model** are:

- Hypothesis $h_\theta \in \mathcal{H}$
- Learning procedure
  - Iterative procedure
  - Cost function



Univariate Linear regression:solving equations

- Hypothesis: $h_\theta(x) = \theta^0 + \theta^1 x$
- Parameters: $\theta = [\theta_0, \theta_1]^T$
- Cost function: $J(\theta_0, \theta_1,) = \frac{1}{2N} \sum_{i=1}^{N} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$
- Goal: $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$
- Iterative method (batch gradient descent)

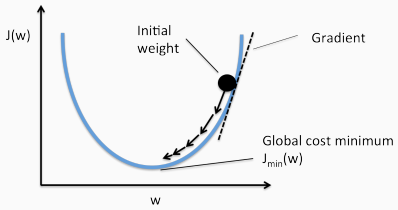
$$\theta_j := \theta_j - \alpha \frac{\partial J}{\partial \theta_j}$$

$$\theta := \theta - \alpha \nabla_\theta J$$

$$\theta_0 := \theta_0 - \alpha \frac{1}{N} \sum_{i=1}^{N} \left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{N} \sum_{i=1}^{N} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x^{(i)}$$

Gradient descent



- Learning parameter: α

Batch vs mini-batch vs stochastic gradient descent

Methods differ in the data points used for each update of the parameters

- Batch gradient descent : (all N data points)
- Mini-batch gradient descent : (random subset of m data points)

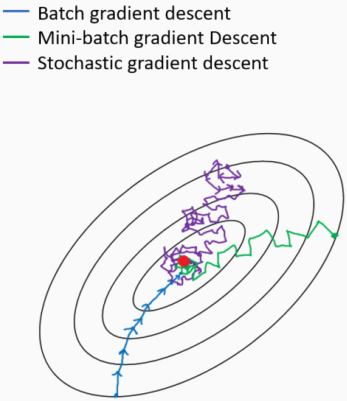
$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x^{(i)}$$

- Stochastic gradient descent : (a single data point *i*, repeat $\forall\, N$)

$$\theta_0 := \theta_0 - \alpha \left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

$$\theta_1 := \theta_1 - \alpha \left(h_\theta(x^{(i)}) - y^{(i)}\right) x^{(i)}$$



Multivariate linear regression

Now we move to one feature x, to a vector of features $\mathbf{x} = \{1, x_1, x_2, \cdots x_N\}$.

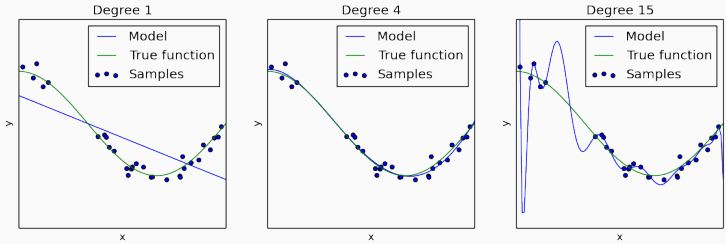
Formally the problem can be framed as follows:

- Hypothesis: $h_{\boldsymbol{\theta}} = \boldsymbol{\theta}^T \mathbf{x}$
- Parameters: $\boldsymbol{\theta} = [\theta_0, \theta_1, \cdots, \theta_N]^T$
- Cost function: $J(\boldsymbol{\theta}) = \frac{1}{2N} \sum_i \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2$
- Goal: $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
- Gradient descent: $\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

The problem of overfitting/underfitting

Here we try to fit our data with polynomial functions of different degree $D = 1, 4, 15$.

This is an example of multivatiate linear regression, with $\mathbf{x} = \{1, x, x^2, \cdots, x^D\}$.



- **Underfitting ($D = 1$)**: too inflexible and cannot capture the pattern
- **Overfitting ($D = 15$)**: too flexible and fits the noise in our data
- **Appropriate fitting ($D = 5$)**: generalization, ability to make good predictions on new data points (not used in the training set).

Regularization

- New cost function with an additional regularization term (Ridge regression):

$$J(\boldsymbol{\theta}) = \frac{1}{2N}\Big[\sum_{i=0}^{N}\big(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)}\big)^2 + \boxed{\lambda\sum_{j=1}^{N}\theta_j^2}\Big]$$

- Calculation of the gradient:

$$\theta_0 := \theta_0 - \alpha(h_\theta(x^{(i)} - y^{(i)}))$$
$$\theta_j := \theta_j(1 - \frac{\alpha\lambda}{N}) - \alpha(h_\theta(x^{(i)} - y^{(i)}))x_j^{(i)}$$

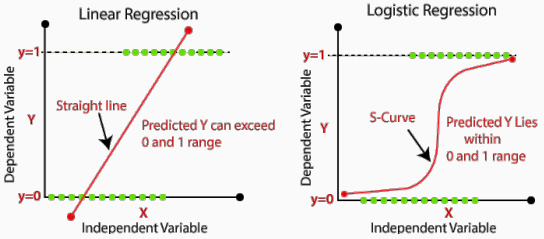
- The strength of the regularization depends on λ:
  - λ small $\Rightarrow$ overfitting
  - λ big $\Rightarrow$ underfitting

- Types of regularization (different regularization term):
  - Ridge
  - Lasso
  - Elastic Net

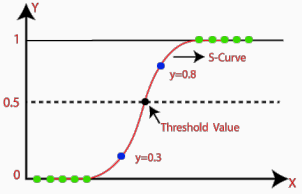
Classification

Logistic regression: intuition

Here we want to predict the correct class for data with binary labels $y^{(i)} \{0, 1\}$



- A straight line is not appropriate to fit these data
- We want a prediction between 0 and 1 that represents the probability to belong to a given class



Logistic regression: solving equations

- Hypothesis (sigmoid function): $h_\theta = [1 + e^{\theta^T \mathbf{x}}]^{-1}$
- Parameters: $\boldsymbol{\theta} = [\theta_0, \theta_1, \cdots, \theta_N]^T$
- Cost function: $J(\boldsymbol{\theta}) = -\frac{1}{N} \sum_i \left(y^{(i)} \cdot log(h_\theta(\mathbf{x}^{(i)})) + (1 - y^{(i)}) \cdot log(1 - h_\theta(\mathbf{x}^{(i)})) \right)$
- Goal: $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
- Gradient descent:

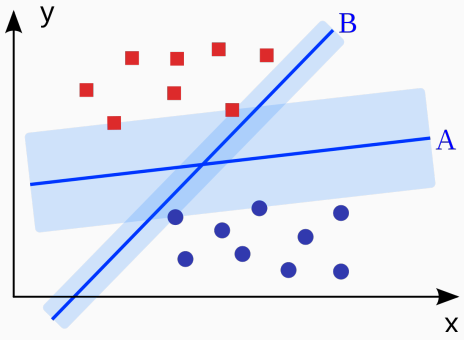
$$\theta_j := \theta_j - \alpha \frac{\partial J(\boldsymbol{\theta})}{\partial \theta}$$

$$\theta_j := \theta_j - \alpha \frac{1}{N} \sum_{i=1}^{N} (h_\theta(\mathbf{x}^{(i)}) - y^{(i)}) x_j^{(i)}$$

Same gradient descend as linear regression but with h_θ sigmoid function

Support vector machine (SVM): intuition

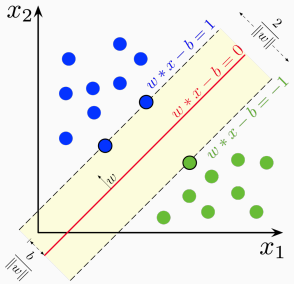
- Multiple hyperplanes could correctly classify binary data
- Can we find the hyperplane that best separates our two classes?



The SVM finds the hyperplane maximizing the margin between our classes

SVM: formal equations

- Our training data are $\{(\mathbf{x}^{(1)}, y^{(1)}), \cdots (\mathbf{x}^{(N)}, y^{(N)})\}$ with $\mathbf{x}^i \in \mathbb{R}^n$ and $y^i = \pm 1$



- We look for an hyperplane: $\mathbf{w}^T\mathbf{x} - b = 0$
- When $y^{(i)} = 1$ the points lie on or above the boundary and when $y^{(i)} = -1$ on or below it:

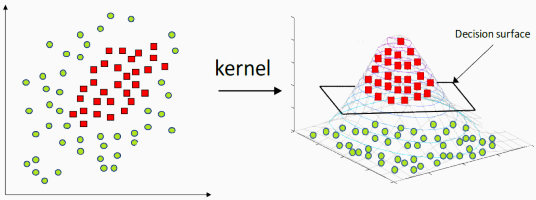
$$\begin{cases} \mathbf{w}^T\mathbf{x}^{(i)} - b \geq 1, & \text{if } y^{(i)} = 1 \\ \mathbf{w}^T\mathbf{x}^{(i)} - b \leq -1, & \text{if } y^{(i)} = -1 \end{cases}$$

- The margin's width is $2/\|w\|$

We want to minimize $\|w\|$ under the constraint $y^{(i)}\,(\mathbf{w}^T\mathbf{x}^{(i)} - b) \geq 1,\ \forall i$

SVM: the kernel trick

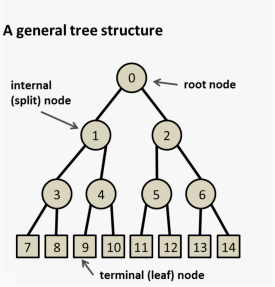
What happen when the data are not linearly separable?

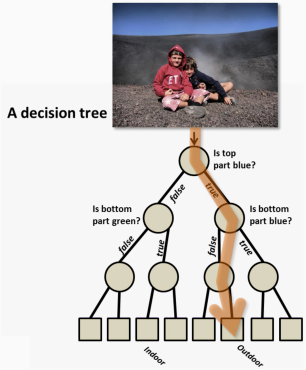


- Proper feature mapping can make non-linear to linear separable
- We do not need the feature space transformation, we only need the kernels
- Popular kernels:
  - Polynomial: $k(x, x') = (\mathbf{x}^T\mathbf{x}' + c)^d$
  - Gaussian: $k(x, x') = exp(-\left\|\mathbf{x} - \mathbf{x}'\right\|^2 / 2\sigma)$

Decision Trees

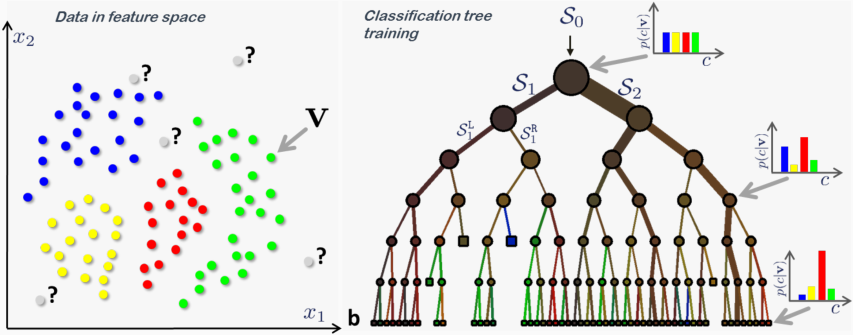
Image classification example





[Criminisi et al, 2011]

Another classification tree



[Criminisi et al, 2011]

How do we build a tree

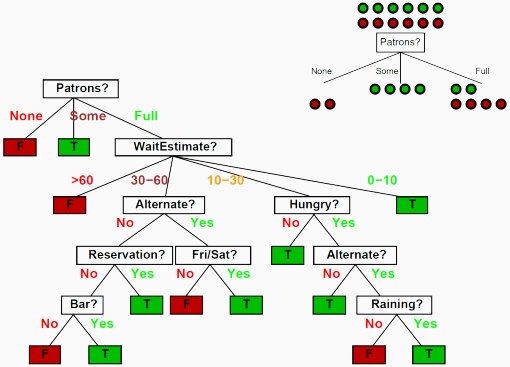
Building a node from data

Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

[AI book of of Stuart Russel and Peter Norvig]

How do we build a tree: A learned tree

A learned tree

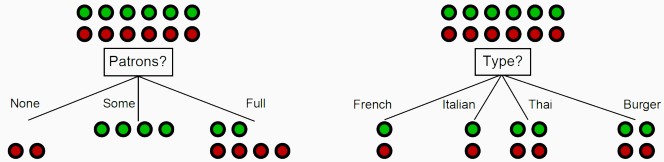


[AI book of of Stuart Russel and Peter Norvig]

How do we build a tree

Which split is better?

Ideally we want to separate negative and positive examples



[AI book of of Stuart Russel and Peter Norvig]

How do we build a tree

We use the information gain as criterion:

- Shannon Entropy

$$H = -\sum_i p_i log_2(p_i)$$

- Expected Entropy (for a feature F with K values)

$$EH(F) = -\sum_{i=1}^{K} \frac{n_i}{N} H_i$$

- Information Gain $I(F)$

$$I(F) = H(Data) - EH(F)$$

How do we build a tree

The patron vs type example

- Entropy data:

$$H(Data) = -2 \cdot \left[1/2 \cdot log_2(1/2)\right] = 1$$

- Entropy left split:

$$EH(L) = -\left[2/12 \cdot 1 \cdot log_2(1) + 4/12 \cdot 1 \cdot log_2(1) + \right.$$
$$\left. + 6/12 \cdot \left(2/6 \cdot log_2(2/6) + 4/6 \cdot log_2(4/6)\right)\right] \approx 0.46$$

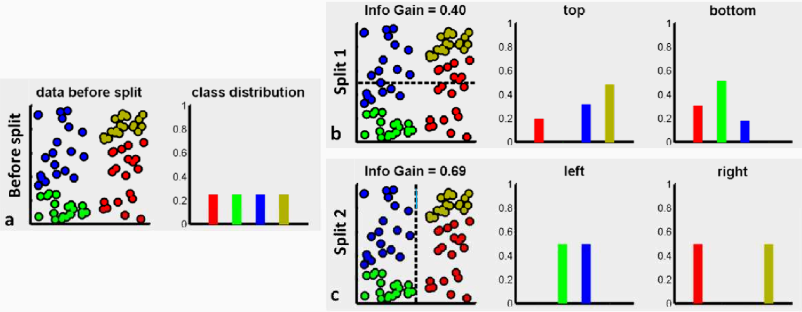
- Entropy right split:

$$EH(R) = -(2/12 + 2/12 + 4/12 + 4/12) \cdot \left[1/2 \cdot log_2(1/2)\right] = 1$$

The information gain is clearly higher in the left split!

How do we build a tree

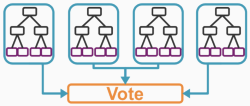
Another simple example



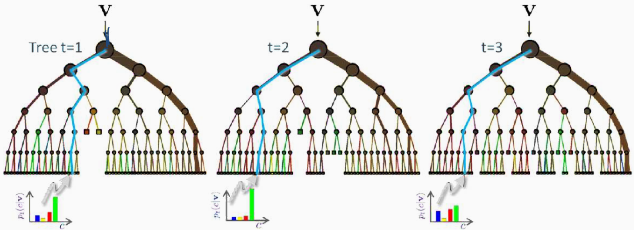
[Criminisi et al, 2011]

Random forest: intuition and example

A multitude of decision trees are generated at training time. The output of the random forest is the class selected by the forest (average or majority vote).



Example: We trained a forest of $T = 3$ trees and we want to classify a new point $\mathbf{v}$.



The new point is classified as green!

[Criminisi et al, 2011]

Random forest algorithm

Given a training set $(\mathbf{x}_1, \cdots, \mathbf{x}_N)$ with responses $(y_1, \cdots, y_N)$:

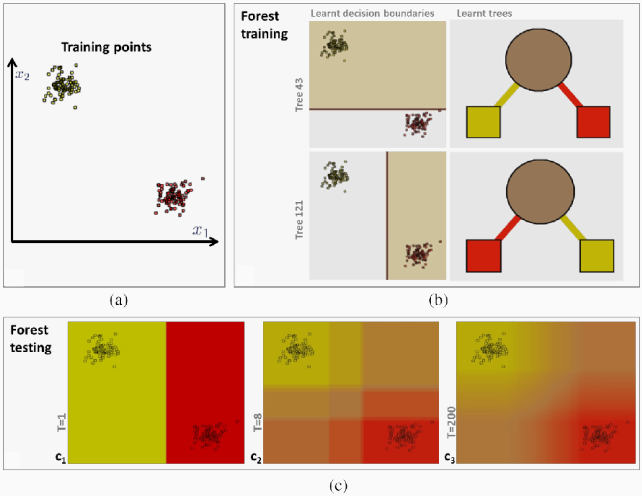
1. For b = 1 to B:
   (a) Draw a bootstrap sample (with replacement) Z of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n min is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point $\mathbf{x}$:

- Let $\hat{C}_b(x)$ be the class prediction of the b-th random-forest tree.
- Then $\hat{C}_{rf}^B(x)$ = majority vote or average of $\{\hat{C}_b(x)\}_1^B$

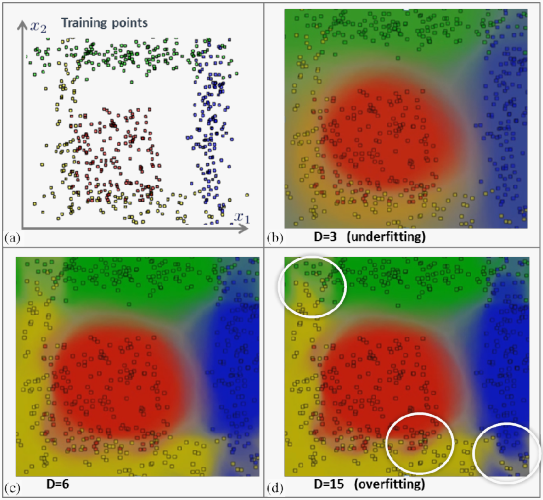
[From the book of Hastie, Friedman and Tibshirani]

Effects of size



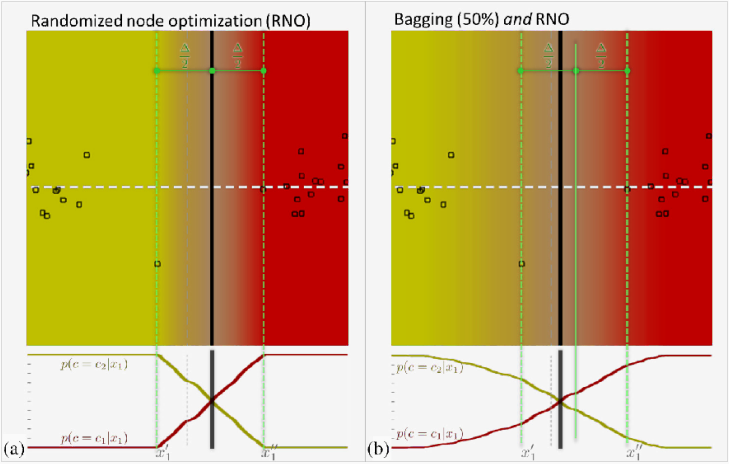
[Criminisi et al, 2011]

Effects of depth



[Criminisi et al, 2011]

Effects of bagging



[Criminisi et al, 2011]

Model evaluation and selection

Measures of performance for a regression model

Mean square error

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(h_\theta(\mathbf{x}^i) - y^i\right)^2$$

Root mean square error

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(h_\theta(\mathbf{x}^i) - y^i\right)^2}$$

Relative square error

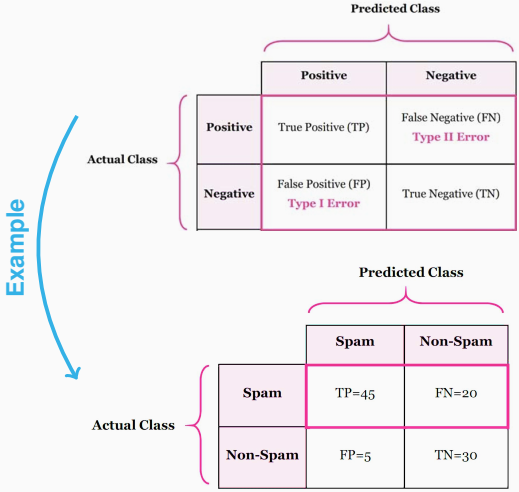
$$RSE = \frac{1}{N}\sum_{i=1}^{N}\frac{\left(h_\theta(\mathbf{x}^i) - y^i\right)^2}{\sum_{i=1}^{N}(y^i - < y >)^2}$$

Coefficient of determination

$$R^2 = 1 - RSE$$

Measure of performance for a classification model

Confusion matrix



Sensitivity/True positive rate

$$TPR = \frac{TP}{TP + FN}$$

Precision/Positive predicted value

$$PPV = \frac{TP}{TP + FP}$$

Specificity/True negative rate

$$TNR = \frac{TN}{FP + TN}$$

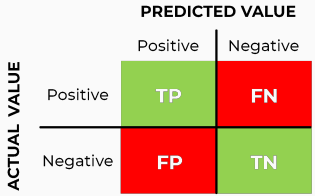
F_1-score

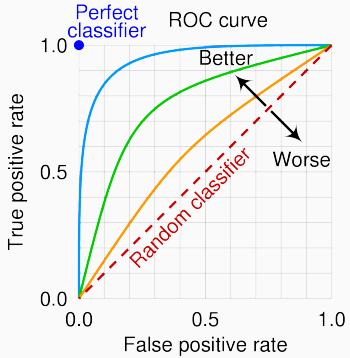
$$F_1 = 2\frac{PPV \cdot TPR}{PPV + TPR}$$

The receiver operating characteristic (ROC) curve

How to construct the ROC curve:

1. Getting model predictions.
2. Calculate the TPR and FPR.
3. Plot TPR and FPR for every cut-off.





$$TPR = \frac{TP}{TP + FN} \qquad TNR = \frac{TN}{FP + TN}$$

Maximizing the area under the curve allows to choose the best model

Model evaluation and selection: training/test sets

Model evaluation

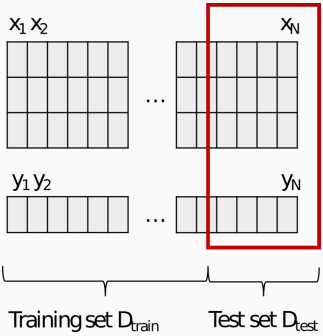
We evaluate a model h by computing the empirical risk ER over the training set D_{train} of size N_{train}

$$ER = \frac{1}{N_{\text{train}}} \sum_{i \in D_{\text{train}}} J(h(x_i), y_i)$$

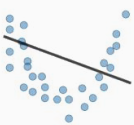
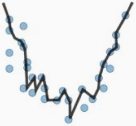
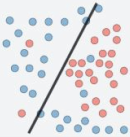
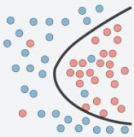
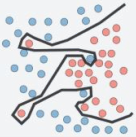
Model selection

For K models $h_1, h_2, \cdots h_K$ we select the best as:

$$h^* = \underset{k=1,\cdots,K}{\operatorname{argmin}} \frac{1}{N_{\text{test}}} \sum_{i \in D_{\text{test}}} J(h_k(x_i), y_i)$$



Training/test set error and appropriate fitting

	Underfitting	Just right	Overfitting
Symptoms	• High training error • Training error close to test error • High bias	• Training error slightly lower than test error	• Very low training error • Training error much lower than test error • High variance
Regression illustration			
Classification illustration			
Deep learning illustration			
Possible remedies	• Complexify model • Add more features • Train longer		• Perform regularization • Get more data

Parameters vs Hyperparameters

In machine learning models there are **two types of "parameters"**:

1. Model Parameters: learned from data automatically via the optimization procedure
2. Model Hyperparameters: set manually or tuned and used in processes to help estimate model parameters

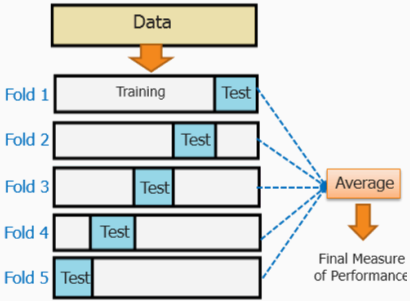
Example of multivariate linear regression with regularization:

- model parameters are values of the vector $\boldsymbol{\theta}$, which are learned from data using the gradient descent algorithm
- model hyperparameters are learning rate α and the regularization parameter λ; both need to be set manually or tuned

K-fold cross validation

K-fold cross validation:

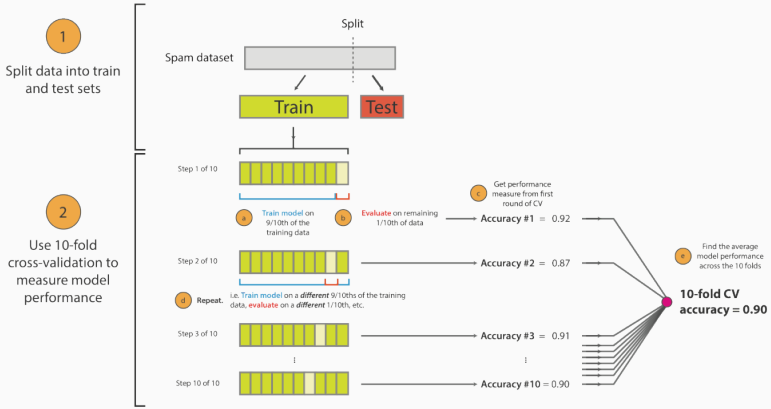
1. Partition the data into K subsets of similar sizes $D_1, \cdots, D_K$
2. For $i = 1, \cdots, K$: train on $D_1, \cdots, D_{i-1}, D_K$
3. At each iteration of the loop evaluate on D_i



(A) Way 1

- Apply k-fold validation on entire data set
- Calculate average of each validation score
- Using it as a final score

K-fold cross validation



(B) Way 2

- Split data into train set, test set
- Apply K-fold validation on train set
- Find best estimator from average of each validation score
- Calculate final score on test set with best estimator

Cross validation

Both ways can be correct depending on what the purpose of the procedure is.

- **(A)** uses cross validation for **validation** (or rather, verification), that is, to estimate generalization error.
- **(B)** uses cross validation for **optimizing some hyperparameters** (e.g. model complexity) and then tests the optimized model with the "test" data.

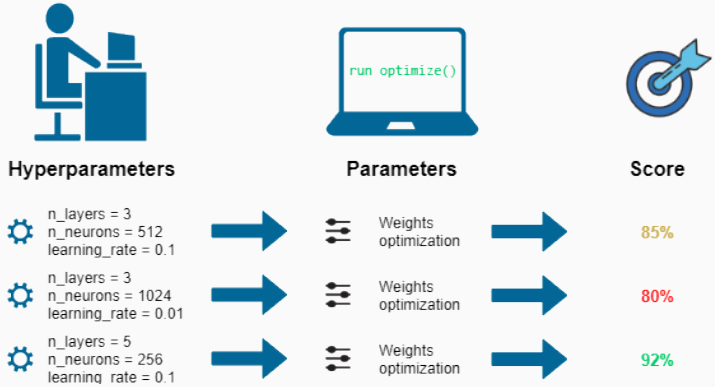
Other usefull remarks:

- The term **"cross validation"** refers only to this particular scheme of splitting data (drawing **without replacement** and calculating a pre-determined number of surrogate models so that each case is used for testing exactly once).
- The **K-fold** procedure is called **leave-one-out when K** $=$ **N**, with N indicating the number of data.
- **Bootstrap** can be alternatively used for validation: similar strategy, except than samples are randomly drawn **with replacement** from the dataset.

Hyperparameters optimization

Hyperparameter tuning is nothing but searching for the right set of hyperparameters to achieve high performance.

Search algorithms must be guided by some performance metric, typically measured by cross-validation on the training set.



Hyperparameter optimization: common methods

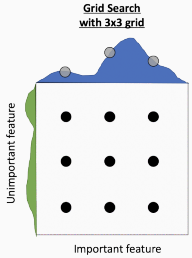
Grid search

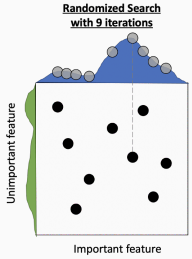
- Search in the cartesian product of these two sets
- The good: simple and parallel. The bad: the cost
- Unfeasible in more than three dimensions

Random search

- Parameters combinations selected randomly.
- The good: cost/parallel. The bad: randomness

The more the dimensions, the more likely random search is to outperform grid search (find optimum faster)





Hyperparameter optimization: more advanced

Bayesian search

- Initial random search strategy
- Performance achieved with prior values is used to seed next choices
- The bad: It is likely to get "sucked into" non optimal solution (when there is a large broad area of moderately-valued points)