Lectures on Machine Learning

Lecture 1: from artificial intelligence to machine learning

Stefano Carrazza

TAE2018, 2-15 September 2018

European Organization for Nuclear Research (CERN)

Acknowledgement: This project has received funding from HICCUP ERC Consolidator grant (614577) and by the European Unions Horizon 2020 research and innovation programme under grant agreement no. 740006.

Why lectures on machine learning?

Why lectures on machine learning?

because

- it is an essential set of algorithms for building models in science,

Why lectures on machine learning?

because

- it is an essential set of algorithms for building models in science,
- fast development of new tools and algorithms in the past years,

Why lectures on machine learning?

because

- it is an essential set of algorithms for building models in science,
- fast development of new tools and algorithms in the past years,
- nowadays it is a requirement in experimental and theoretical physics,

Why lectures on machine learning?

because

- it is an essential set of algorithms for building models in science,
- fast development of new tools and algorithms in the past years,
- nowadays it is a requirement in experimental and theoretical physics,
- large interest from the HEP community: *IML*, *conferences*, *grants*.

What expect from these lectures?

What expect from these lectures?

- Learn the basis of machine learning techniques.
- Learn when and how to apply machine learning algorithms.

The talk is divided in three lectures:

Lecture 1 (today)

- Artificial intelligence
- Machine learning
- Model representation
- Metrics

Lecture 2 (tomorrow)

- Parameter learning
- Non-linear models
- Beyond neural networks
- Clustering

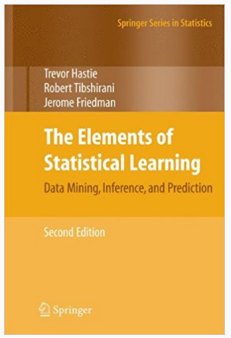
Lecture 3 (tomorrow)

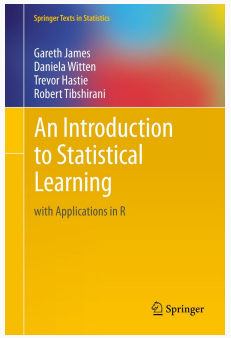
- Hyperparameter tune
- Cross-validation
- ML in practice
- The PDF case study

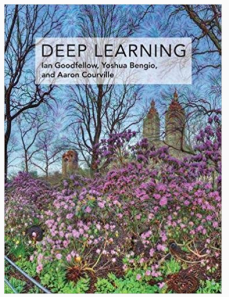
Some references

Books:

- *The elements of statistical learning*, T. Hastie, R. Tibshirani, J. Friedman.
- *An introduction to statistical learning*, G. James, D. Witten, T. Hastie, R. Tibshirani.
- *Deep learning*, I. Goodfellow, Y. Bengio, A. Courville.





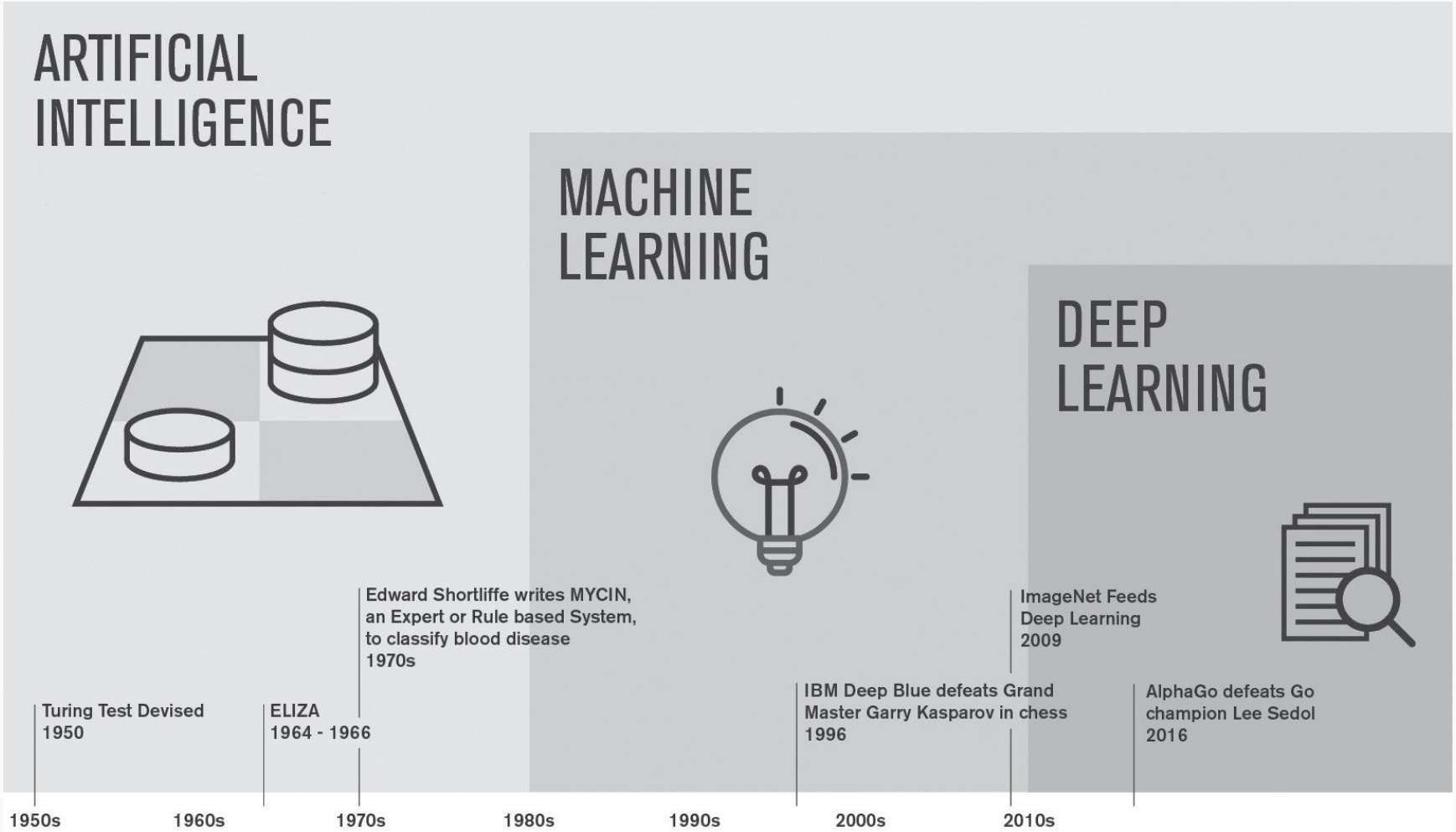


Online resources:

- HEP-ML: `https://github.com/iml-wg/HEP-ML-Resources`
- Tensorflow: `http://tensorflow.org`
- Keras: `http://keras.io`
- Scikit: `http://scikit-learn.org`

Artificial Intelligence

Artificial intelligence timeline

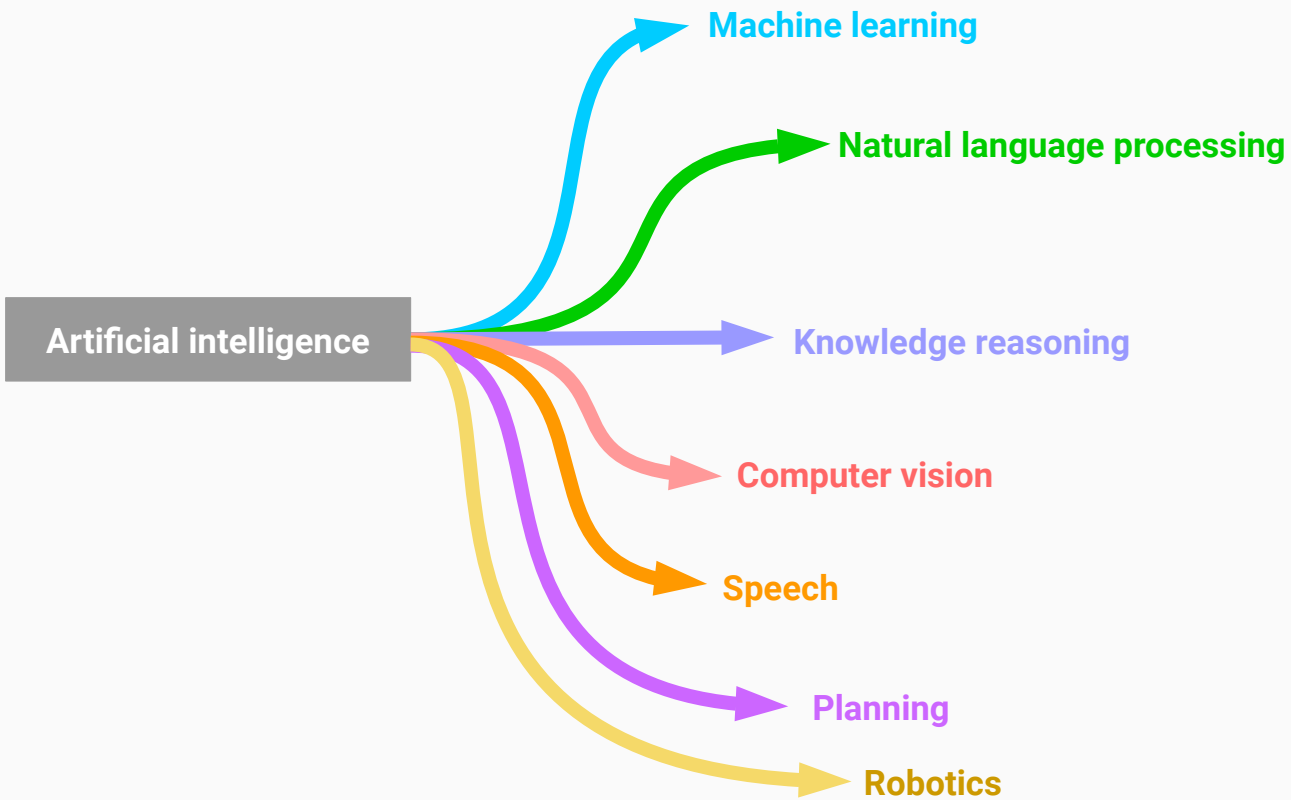


Defining A.I.

Artificial intelligence (A.I.) is *the science and engineering of making intelligent machines*. (John McCarthy '56)

Defining A.I.

Artificial intelligence (A.I.) is *the science and engineering of making intelligent machines*. (John McCarthy '56)

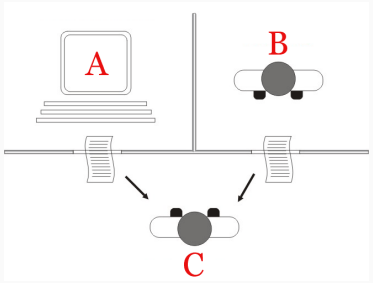


A.I. consist in the development of computer systems to perform tasks commonly associated with intelligence, such as *learning*.

A.I. and humans

There are two categories of A.I. tasks:

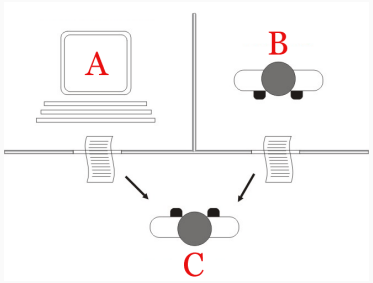
- **abstract and formal:** easy for computers but difficult for humans, *e.g.* play chess (IBM's Deep Blue 1997).
  $\rightarrow$ *Knowledge-based* approach to artificial intelligence.



A.I. and humans

There are two categories of A.I. tasks:

- **abstract and formal:** easy for computers but difficult for humans, *e.g.* play chess (IBM's Deep Blue 1997).
  → *Knowledge-based* approach to artificial intelligence.



- **intuitive for humans but hard to describe formally:**
  e.g. recognizing faces in images or spoken words.
  → *Concept* capture and generalization

A.I. technologies

Historically, the *knowledge-based* approach has not led to a major success with intuitive tasks for humans, because:

- requires human *supervision* and hard-coded *logical inference rules*.
- lacks of *representation learning* ability.

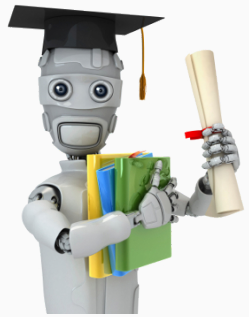
A.I. technologies

Historically, the *knowledge-based* approach has not led to a major success with intuitive tasks for humans, because:

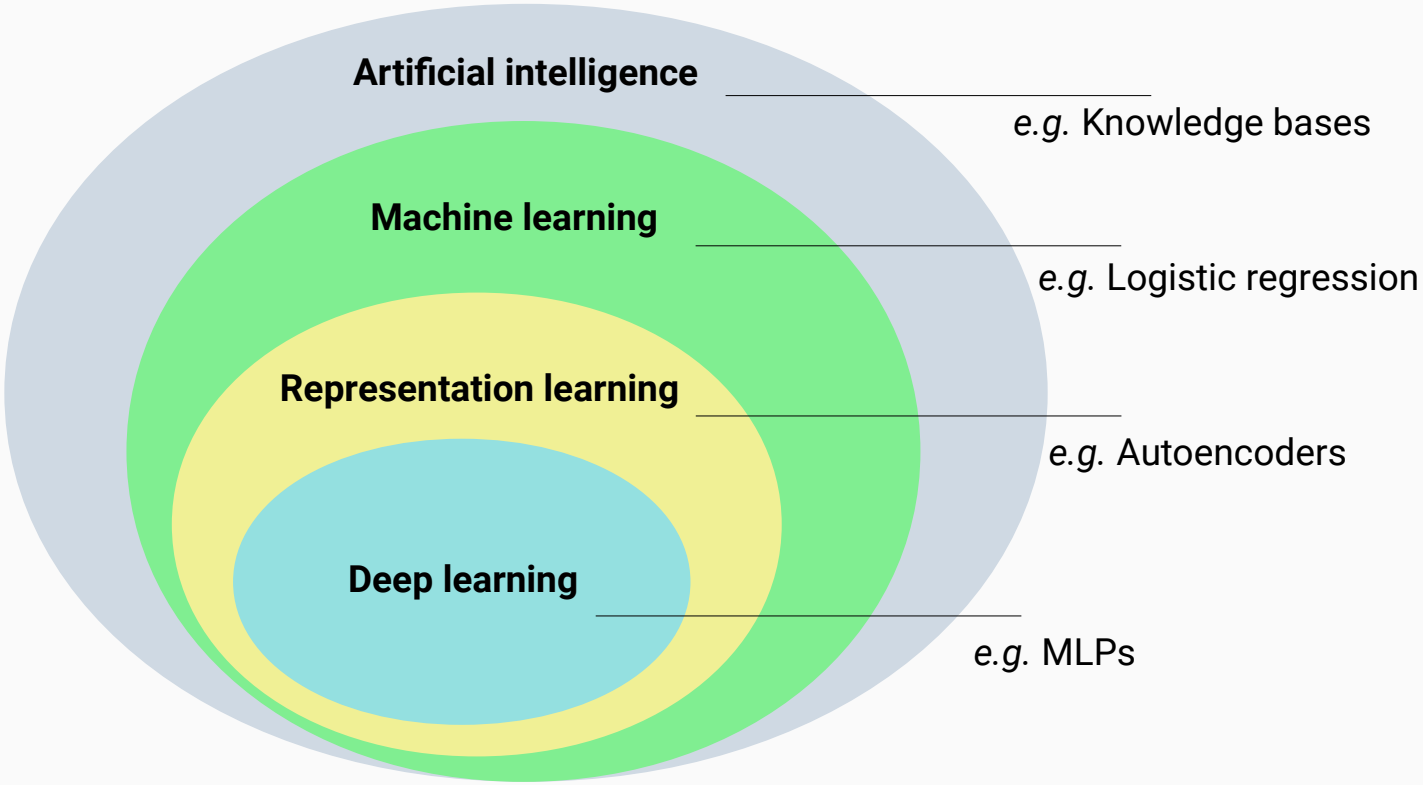
- requires human *supervision* and hard-coded *logical inference rules*.
- lacks of *representation learning* ability.

Solution:

The A.I. system needs to *acquire its own knowledge*.
This capability is known as **machine learning** (ML).
$\rightarrow$ *e.g.* write a program which learns the task.



Venn diagram for A.I.



When a representation learning is difficult, ML provides **deep learning** techniques which allow the computer to build complex concepts out of simpler concepts, *e.g.* artificial neural networks (MLP).

Machine Learning

Machine learning definition

Definition from A. Samuel in 1959:

Field of study that gives computers the ability to learn without being explicitly programmed.

Machine learning definition

Definition from A. Samuel in 1959:

Field of study that gives computers the ability to learn without being explicitly programmed.

Definition from T. Mitchell in 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 on T, as measured by P, improves with experience E.

Machine learning examples

Thanks to work in A.I. and new capability for computers:

- **Database mining:**
  - Search engines
  - Spam filters
  - Medical and biological records

Machine learning examples

Thanks to work in A.I. and new capability for computers:

- **Database mining:**
  - Search engines
  - Spam filters
  - Medical and biological records
- **Intuitive tasks for humans:**
  - Autonomous driving
  - Natural language processing
  - Robotics (reinforcement learning)
  - Game playing (DQN algorithms)

Machine learning examples

Thanks to work in A.I. and new capability for computers:

- **Database mining:**
  - Search engines
  - Spam filters
  - Medical and biological records
- **Intuitive tasks for humans:**
  - Autonomous driving
  - Natural language processing
  - Robotics (reinforcement learning)
  - Game playing (DQN algorithms)

Machine learning examples

Thanks to work in A.I. and new capability for computers:

- **Database mining:**
  - Search engines
  - Spam filters
  - Medical and biological records
- **Intuitive tasks for humans:**
  - Autonomous driving
  - Natural language processing
  - Robotics (reinforcement learning)
  - Game playing (DQN algorithms)
- **Human learning:**
  - Concept/human recognition
  - Computer vision
  - Product recommendation

ML applications in HEP

ML in experimental HEP

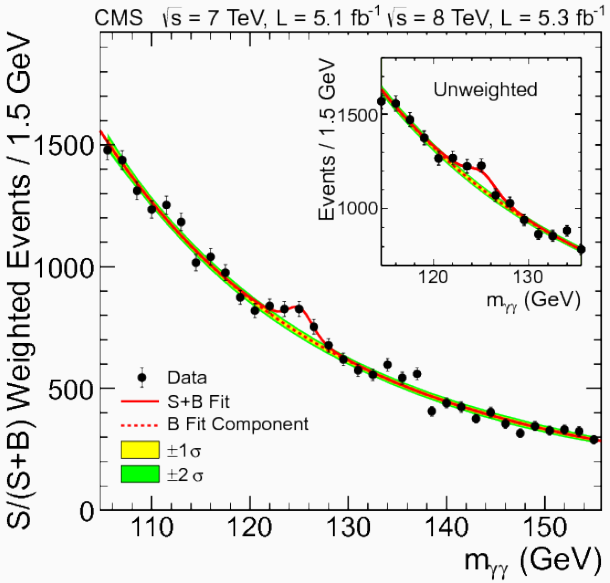
There are many applications in experimental HEP involving the **LHC measurements**, including the **Higgs discovery**, such as:

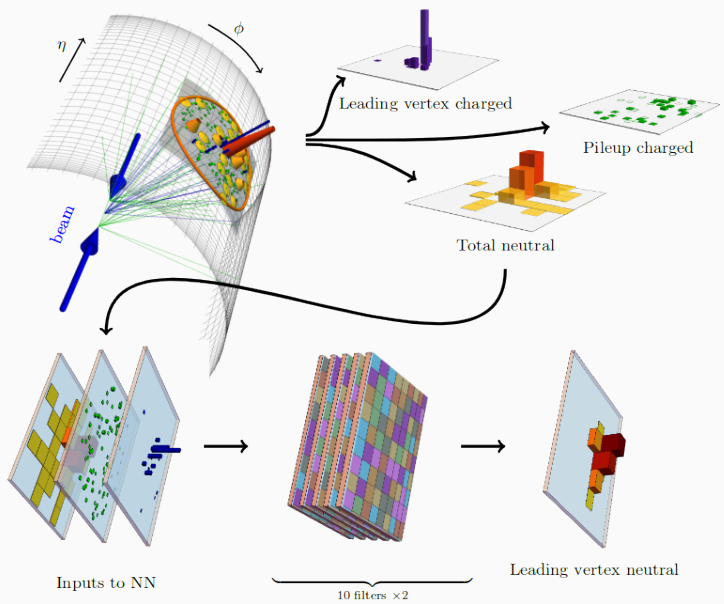
- Tracking
- Fast Simulation
- Particle identification
- Event filtering

ML in experimental HEP

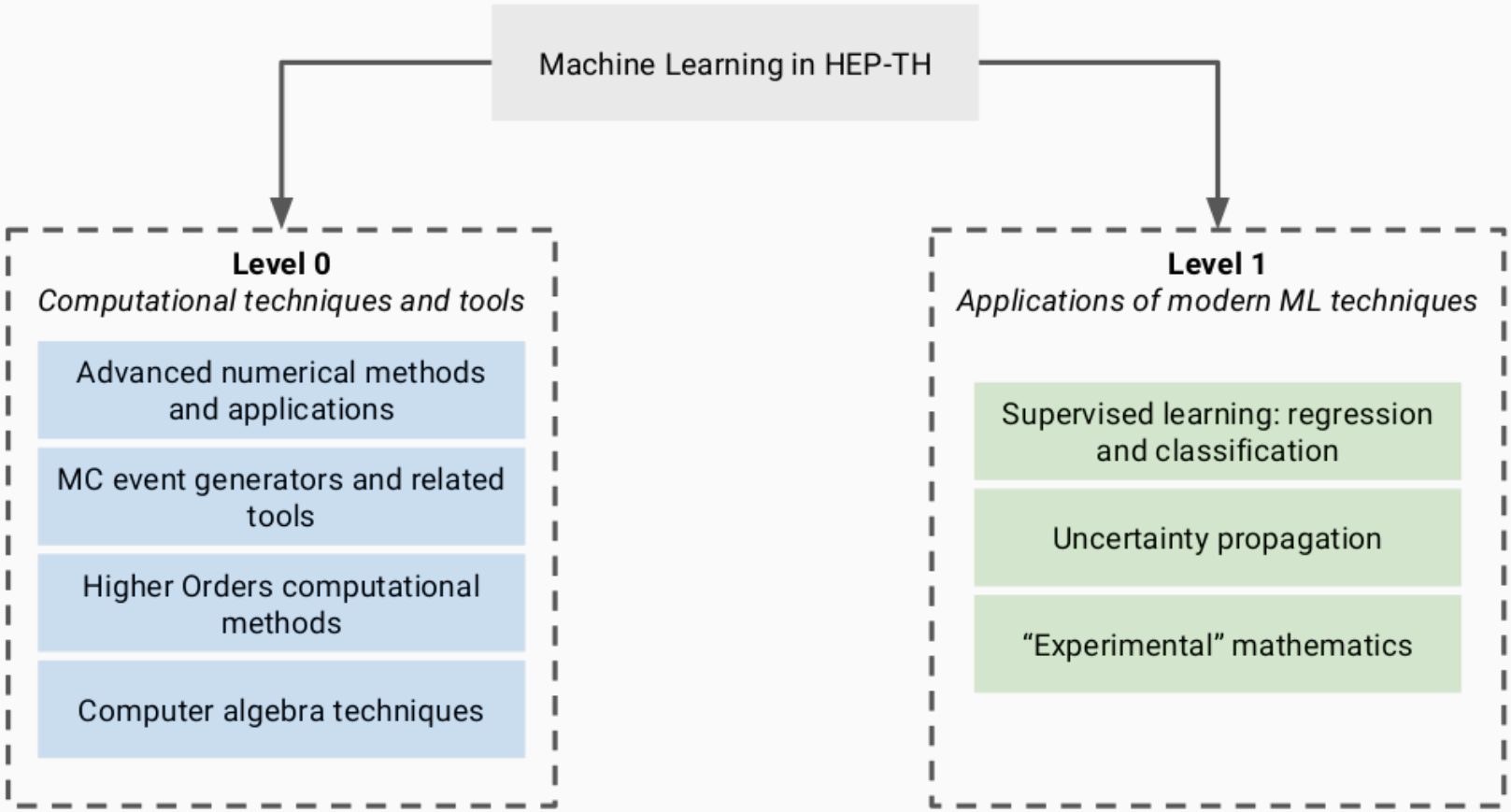
Some remarkable examples are:

- **Signal-background detection:**
  Decision trees, artificial neural networks, support vector machines.
- **Jet discrimination:**
  Deep learning imaging techniques via convolutional neural networks.
- **HEP detector simulation:**
  Generative adversarial networks, *e.g.* LAGAN and CaloGAN.



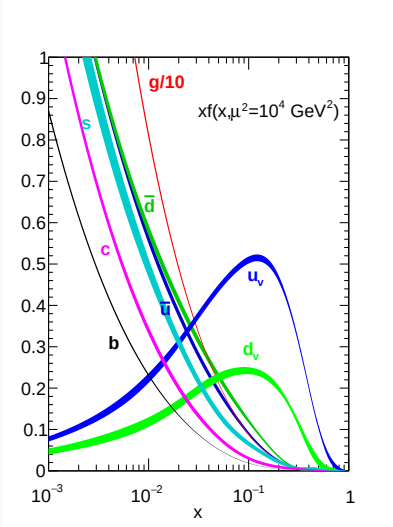


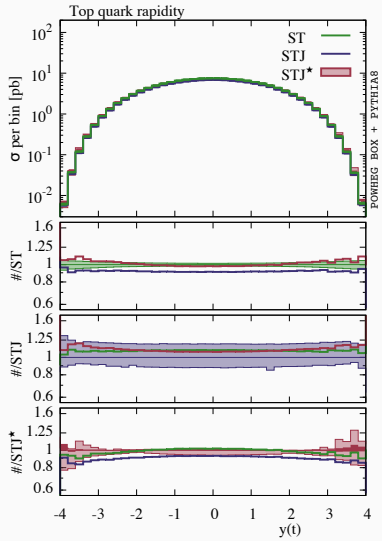
ML in theoretical HEP



ML in theoretical HEP

- **Supervised learning:**
  - The structure of the proton at the LHC
    - parton distribution functions
  - Theoretical prediction and combination
  - Monte Carlo reweighting techniques
    - neural network Sudakov
  - BSM searches and exclusion limits
- **Unsupervised learning:**
  - Clustering and compression
    - PDF4LHC15 recommendation
  - Density estimation and anomaly detection
    - Monte Carlo sampling

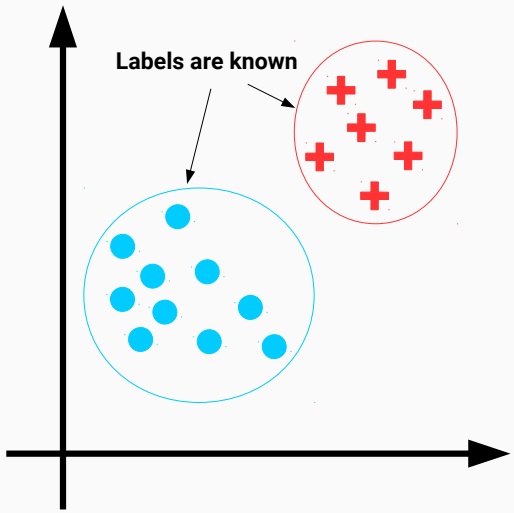


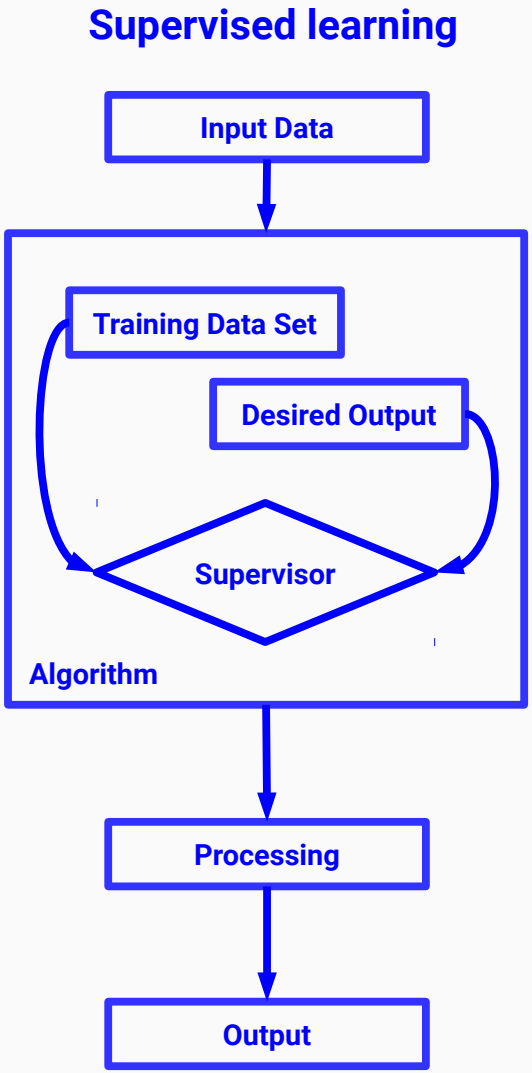


Machine learning algorithms

Machine learning algorithms:

- Supervised learning:
  regression, classification, ...

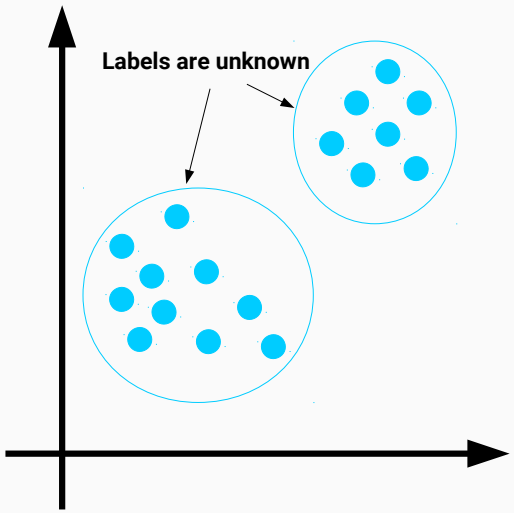


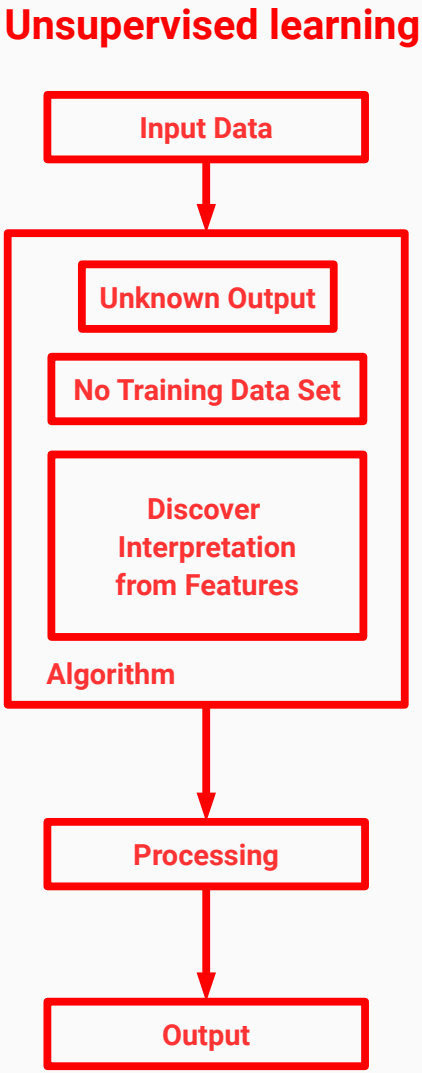


Machine learning algorithms

Machine learning algorithms:

- Supervised learning:
  regression, classification, ...
- Unsupervised learning:
  clustering, dim-reduction, ...

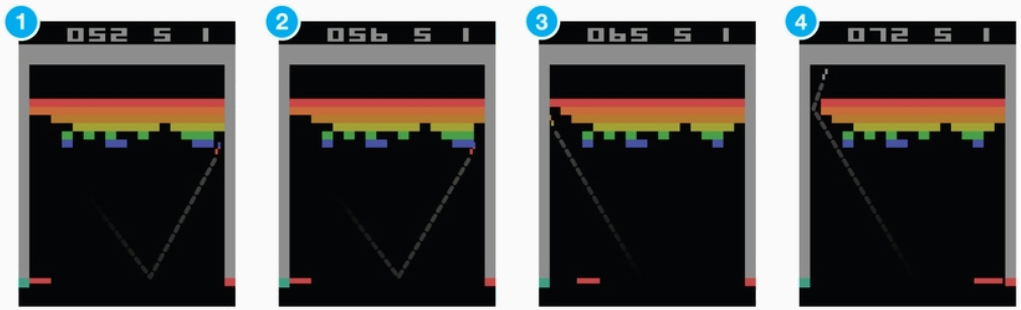




Machine learning algorithms

Machine learning algorithms:

- Supervised learning:
  regression, classification, ...
- Unsupervised learning:
  clustering, dim-reduction, ...
- Reinforcement learning:
  real-time decisions, ...

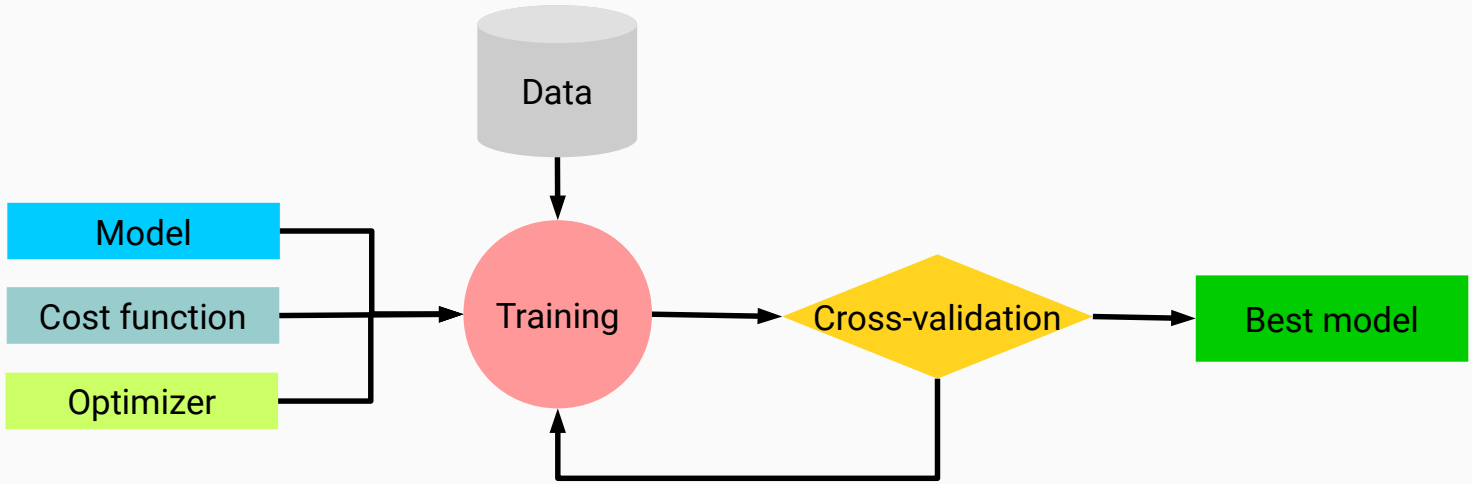


Machine learning algorithms

More than 60 algorithms.

Workflow in machine learning

The operative workflow in ML is summarized by the following steps:

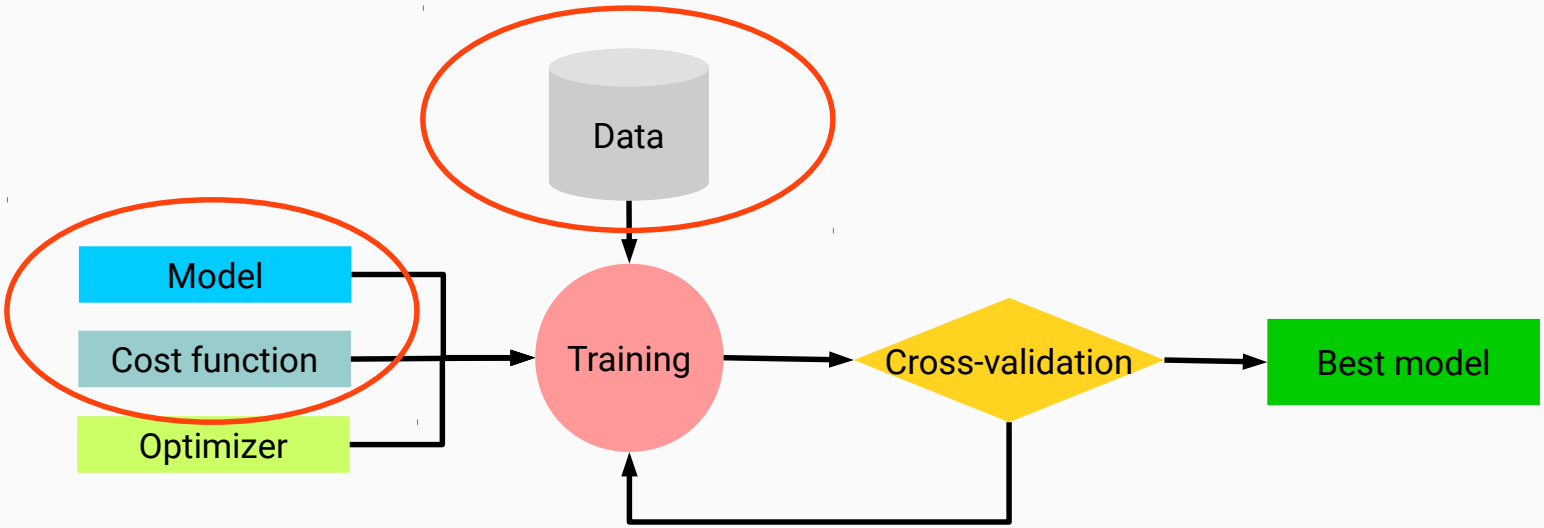


The best model is then used to:

- supervised learning: make predictions for new observed data.
- unsupervised learning: extract features from the input data.

Models and metrics

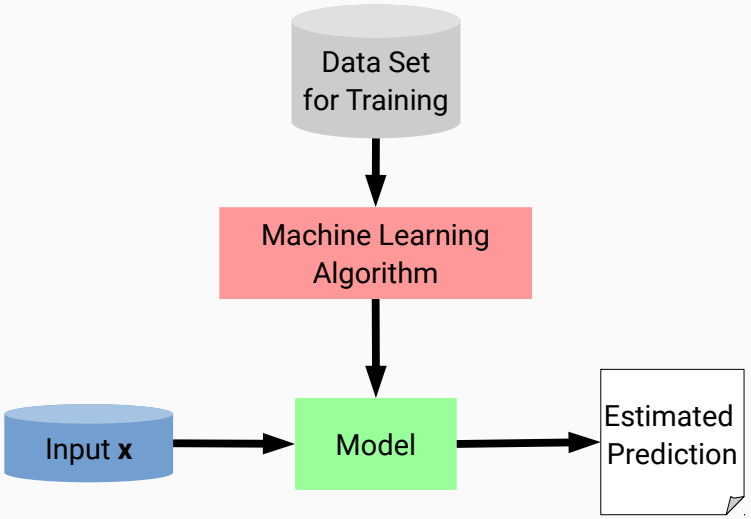
Models and metrics



Model representation in supervised learning

We define parametric and structure models for statistical inference:

- **examples:** linear models, neural networks, decision tree...



- Given a training set of input-output pairs $A = (x_1, y_1), \ldots, (x_n, y_n)$.
- Find a model $\mathcal{M}$ which:

$$\mathcal{M}(\boldsymbol{x}) \sim \boldsymbol{y}$$

where $\boldsymbol{x}$ is the input vector and $\boldsymbol{y}$ discrete labels in classification and real values in regression.

Model representation in supervised learning

Examples of models:

$\rightarrow$ linear regression we define a vector $\boldsymbol{x} \in \mathbb{R}^n$ as input and predict the value of a scalar $y \in \mathbb{R}$ as its output:

$$\hat{y}(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b$$

where $\boldsymbol{w} \in \mathbb{R}^n$ is a vector of parameters and b a constant.

Model representation in supervised learning

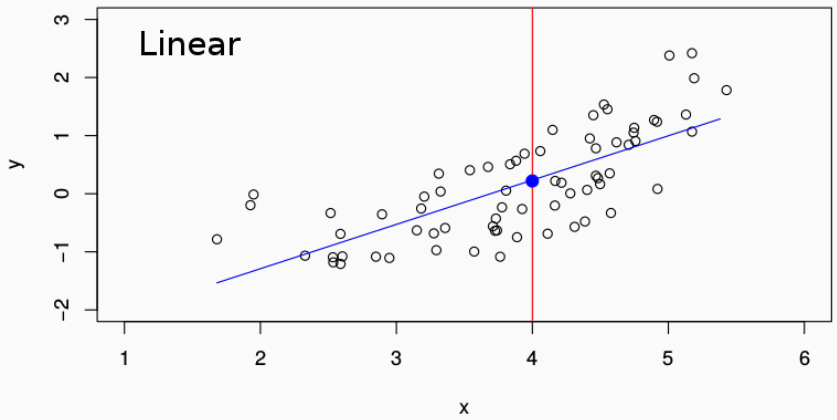
Examples of models:

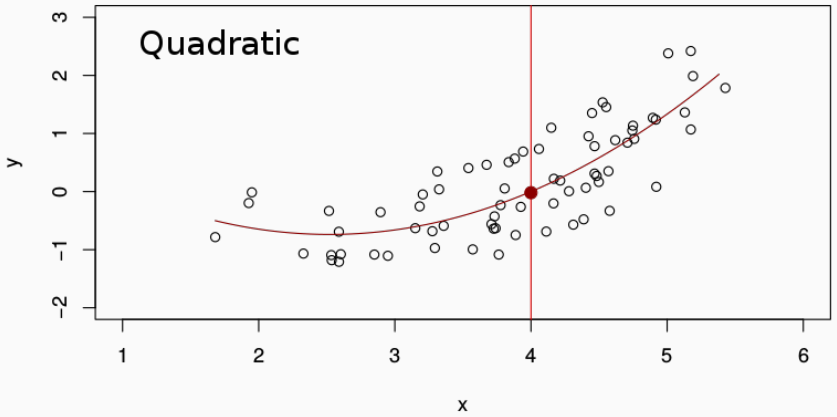
$\rightarrow$ linear regression we define a vector $\boldsymbol{x} \in \mathbb{R}^n$ as input and predict the value of a scalar $y \in \mathbb{R}$ as its output:

$$\hat{y}(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b$$

where $\boldsymbol{w} \in \mathbb{R}^n$ is a vector of parameters and b a constant.

$\rightarrow$ Generalized linear models are also available increasing the power of linear models:





Model representation in supervised learning

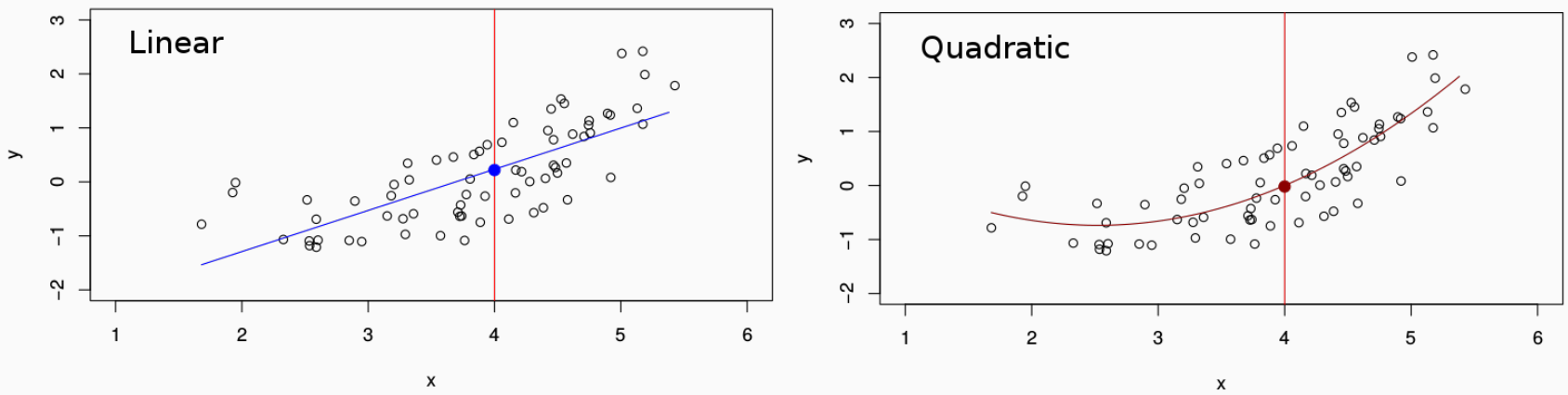
Examples of models:

$\rightarrow$ linear regression we define a vector $\boldsymbol{x} \in \mathbb{R}^n$ as input and predict the value of a scalar $y \in \mathbb{R}$ as its output:

$$\hat{y}(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b$$

where $\boldsymbol{w} \in \mathbb{R}^n$ is a vector of parameters and b a constant.

$\rightarrow$ Generalized linear models are also available increasing the power of linear models:

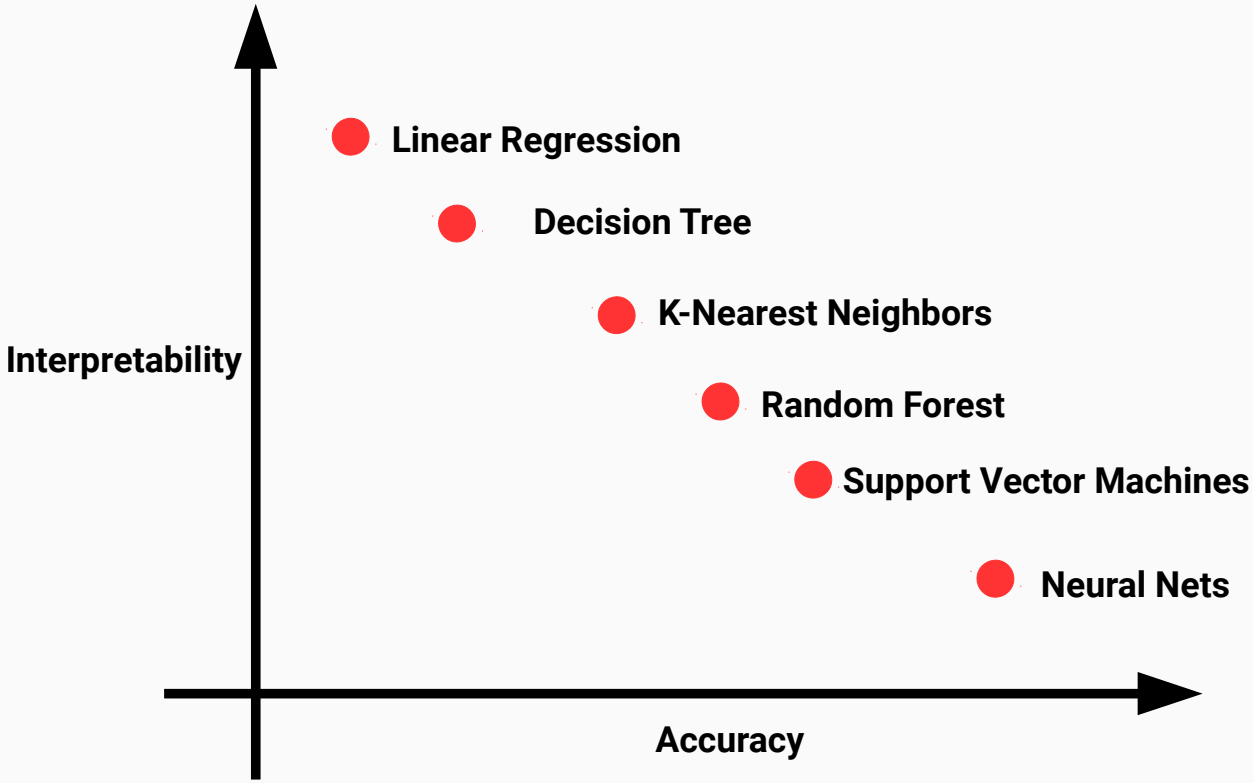


$\rightarrow$ Non-linear models: neural networks (talk later).

Model representation trade-offs

However, the selection of the appropriate model comes with trade-offs:

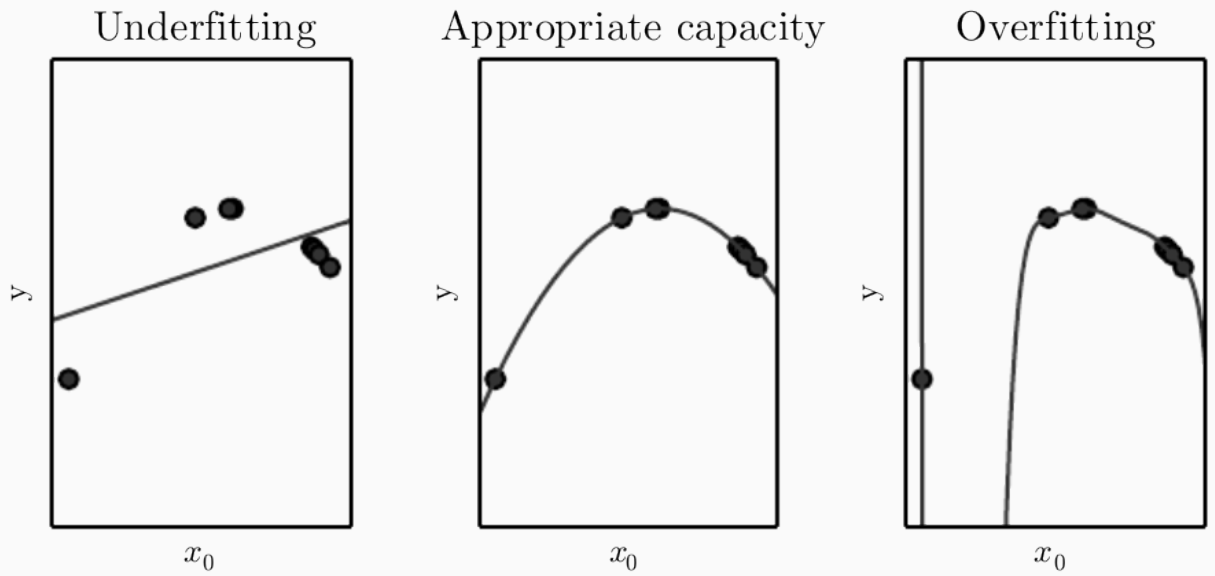
- **Prediction accuracy vs interpretability**:
  $\rightarrow$ *e.g.* linear model vs splines or neural networks.



Model representation trade-offs

However, the selection of the appropriate model comes with trade-offs:

- **Prediction accuracy vs interpretability**:
  $\rightarrow$ *e.g.* linear model vs splines or neural networks.
- **Optimal capacity/flexibility:** number of parameters, architecture
  $\rightarrow$ deal with overfitting, and underfitting situations



Assessing the model performance

How to check model performance?

$\rightarrow$ define metrics and statistical estimators for model performance.

Examples:

- Regression: cost / loss / error function,
- Classification: cost function, precision, accuracy, recall, ROC, AUC

Assessing the model performance - cost function

To access the model performance we define a cost function $J(\boldsymbol{w})$ which often measures the difference between the target and the model output.

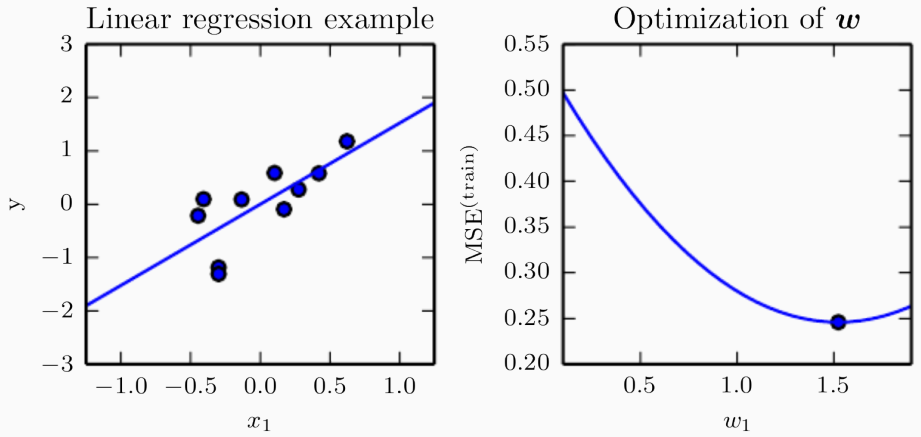
In a optimization procedure, given a model $\hat{y}_{\boldsymbol{w}}$, we search for:

$$\arg\min_{\boldsymbol{w}} J(\boldsymbol{w})$$

The mean square error (MSE) is the most commonly used for regression:

$$J(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_{\boldsymbol{w}}(\boldsymbol{x}_i))^2$$

a quadratic function and convex function in linear regression.



Assessing the model performance - cost function

Other cost functions are depending on the nature of the problem.

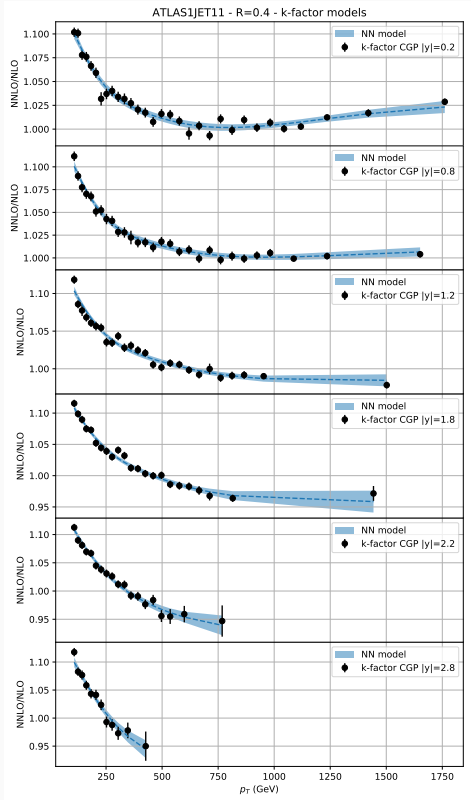
Some other examples:

- regression with uncertainties, chi-square:

$$J(\boldsymbol{w}) = \sum_{i,j=1}^{n} (y_i - \hat{y}_{\boldsymbol{w}}(\boldsymbol{x}_i))(\sigma^{-1})_{ij}(y_j - \hat{y}_{\boldsymbol{w}}(\boldsymbol{x}_j))$$

  where:

  - σ_{ij} is the data covariance matrix.
    e.g. for LHC data experimental statistical and systematics correlations.

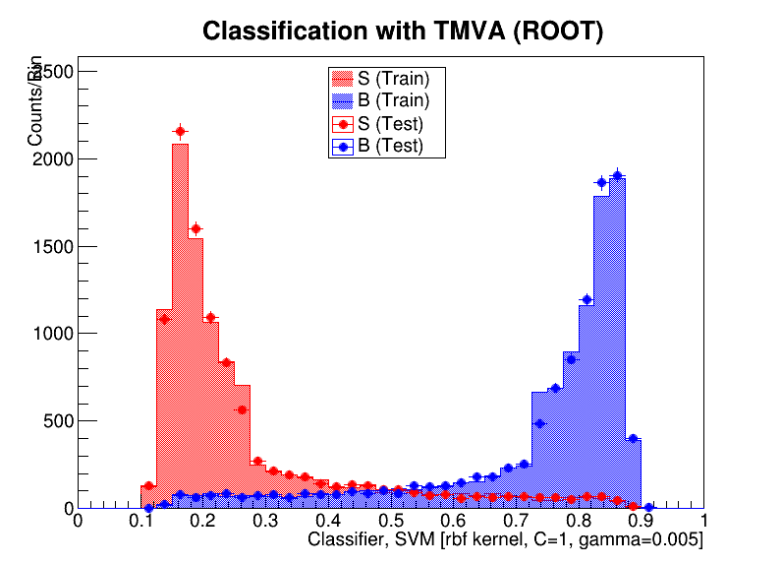


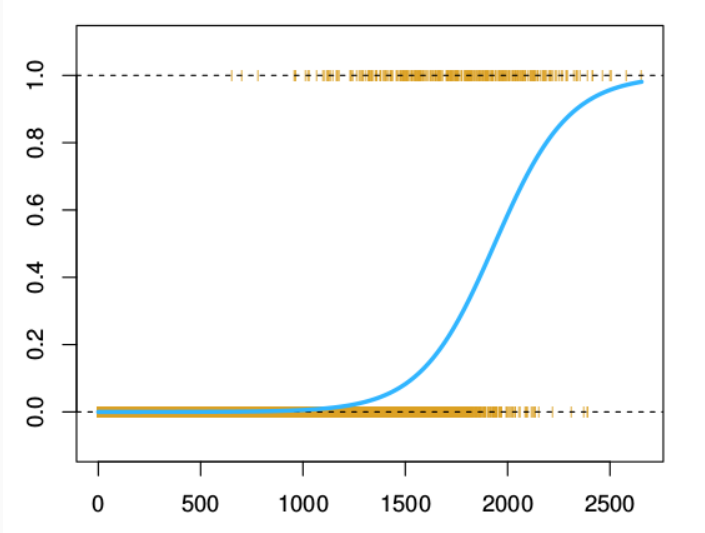
Assessing the model performance - cost function

- logistic regression (binary classification): cross-entropy

$$J(\boldsymbol{w}) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log \hat{y}_{\boldsymbol{w}}(\boldsymbol{x}_i) + (1-y_i)\log(1-\hat{y}_{\boldsymbol{w}}(\boldsymbol{x}_i))$$

where $\hat{y}_{\boldsymbol{w}}(\boldsymbol{x}_i) = 1/(1+e^{-\boldsymbol{w}^T\boldsymbol{x}_i})$.

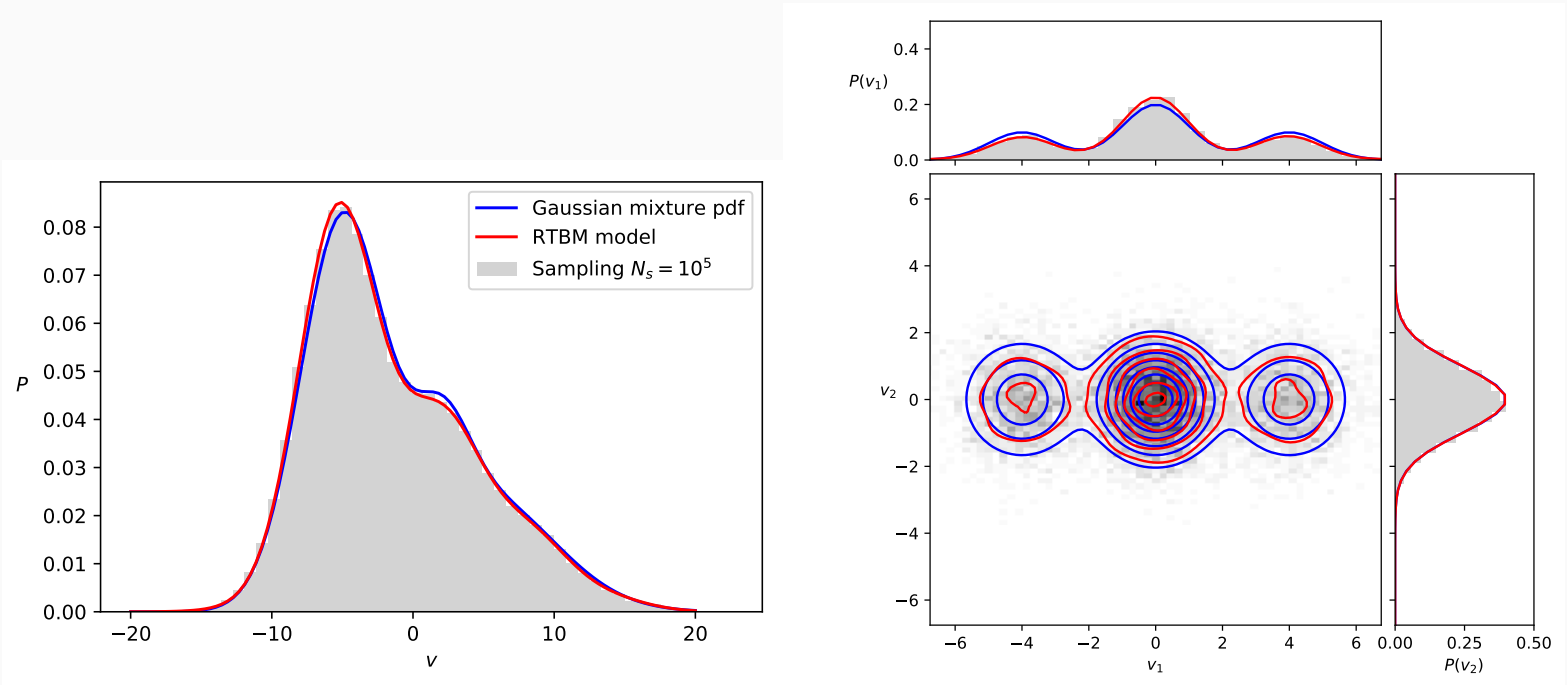




Assessing the model performance - cost function

- density estimate / regression: negative log-likelihood:

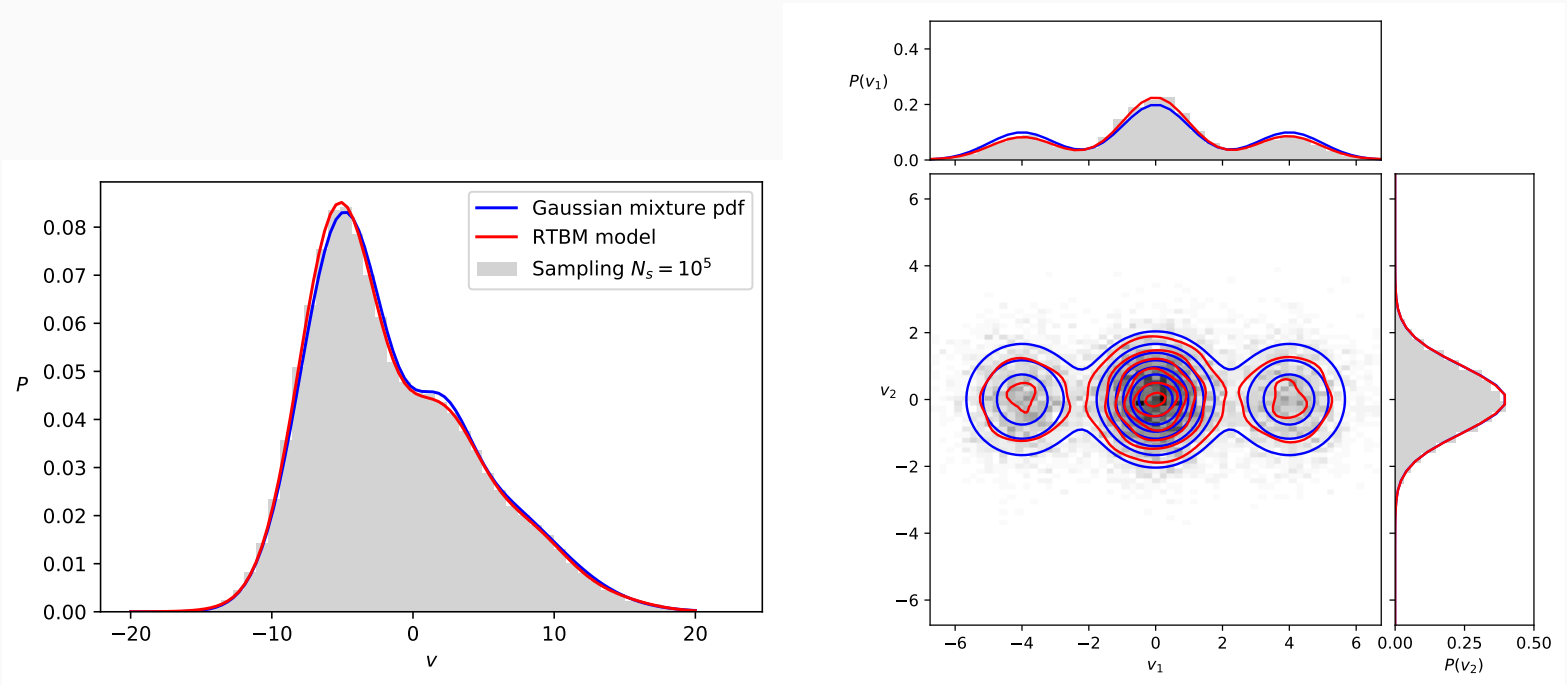
$$J(\boldsymbol{w}) = -\sum_{i=1}^{n} \log(\hat{y}_{\boldsymbol{w}}(\boldsymbol{x}_i))$$



Assessing the model performance - cost function

- density estimate / regression: negative log-likelihood:

$$J(\boldsymbol{w}) = -\sum_{i=1}^{n} \log(\hat{y}_{\boldsymbol{w}}(\boldsymbol{x}_i))$$



- Kullback-Leibler, RMSE, MAE, etc.

Training and test sets

Another common issue related to model capacity in supervised learning:

- The model should not learn **noise** from data.
- The model should be able to **generalize** its output to new samples.

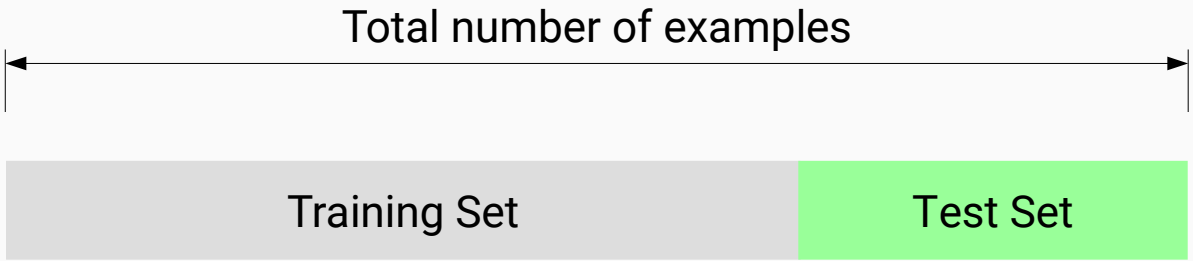
Training and test sets

Another common issue related to model capacity in supervised learning:

- The model should not learn **noise** from data.
- The model should be able to **generalize** its output to new samples.

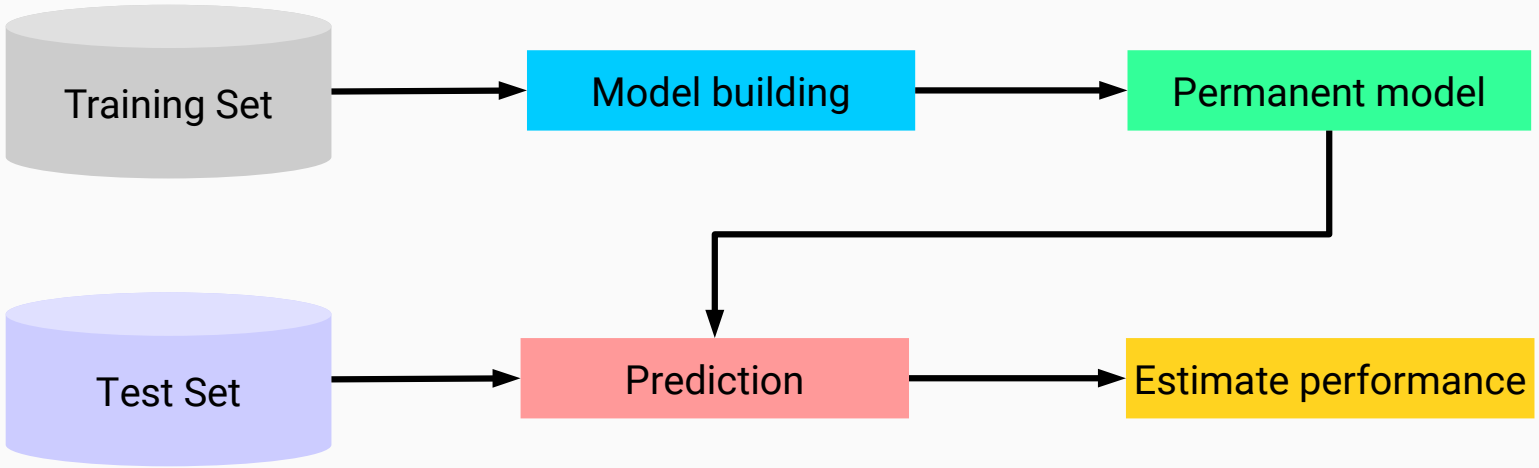
To observe this issue we split the input data in training and test sets:

- training set error, $J_{\text{Tr}}(\boldsymbol{w})$
- test set/generalization error, $J_{\text{Test}}(\boldsymbol{w})$



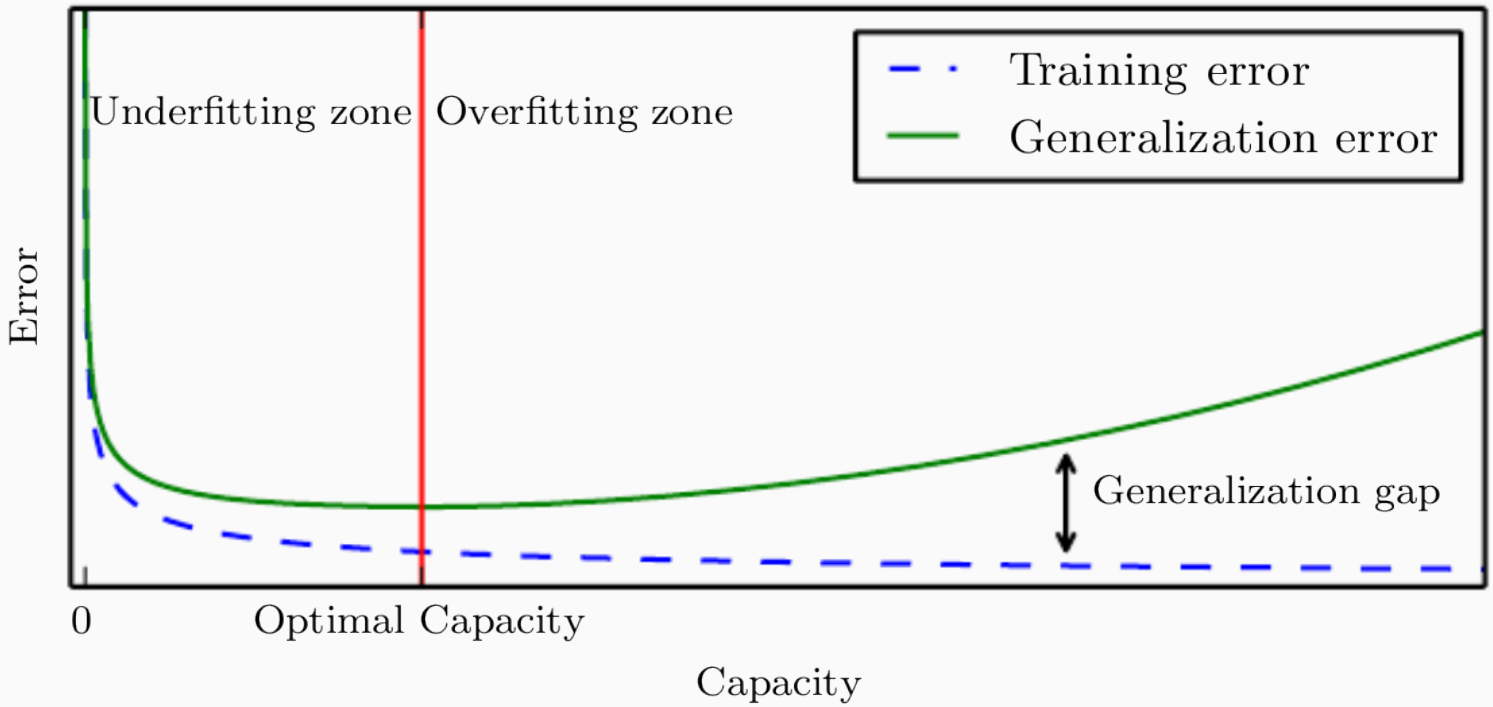
Training and test sets

The test set is independent from the training set but follows the same probability distribution.



Bias-variance trade-off

From a practical point of view dividing the input data in training and test:



The training and test/generalization error conflict is known as bias-variance trade-off.

Bias-variance trade-off

Supposing we have model $\hat{y}(\boldsymbol{x})$ determined from a training data set, and considering as the true model

$$Y = y(X) + \epsilon, \text{ with } y(x) = E(Y|X = x),$$

where the noise ϵ has zero mean and constant variance.

Bias-variance trade-off

Supposing we have model $\hat{y}(\boldsymbol{x})$ determined from a training data set, and considering as the true model

$$Y = y(X) + \epsilon, \text{ with } y(x) = E(Y|X = x),$$

where the noise ϵ has zero mean and constant variance.

If we take (x_0, y_0) from the test set then:

$$E[(y_0 - \hat{y}(x_0))^2] = (\text{Bias}[\hat{y}(x_0)])^2 + \text{Var}[\hat{y}(x_0)] + \text{Var}(\epsilon),$$

where

- $\text{Bias}[\hat{y}(x_0)] = E[\hat{y}(x_0)] - y(x_0)$
- $\text{Var}[\hat{y}(x_0)] = E[\hat{y}(x_0)^2] - (E[\hat{y}(x_0)])^2$

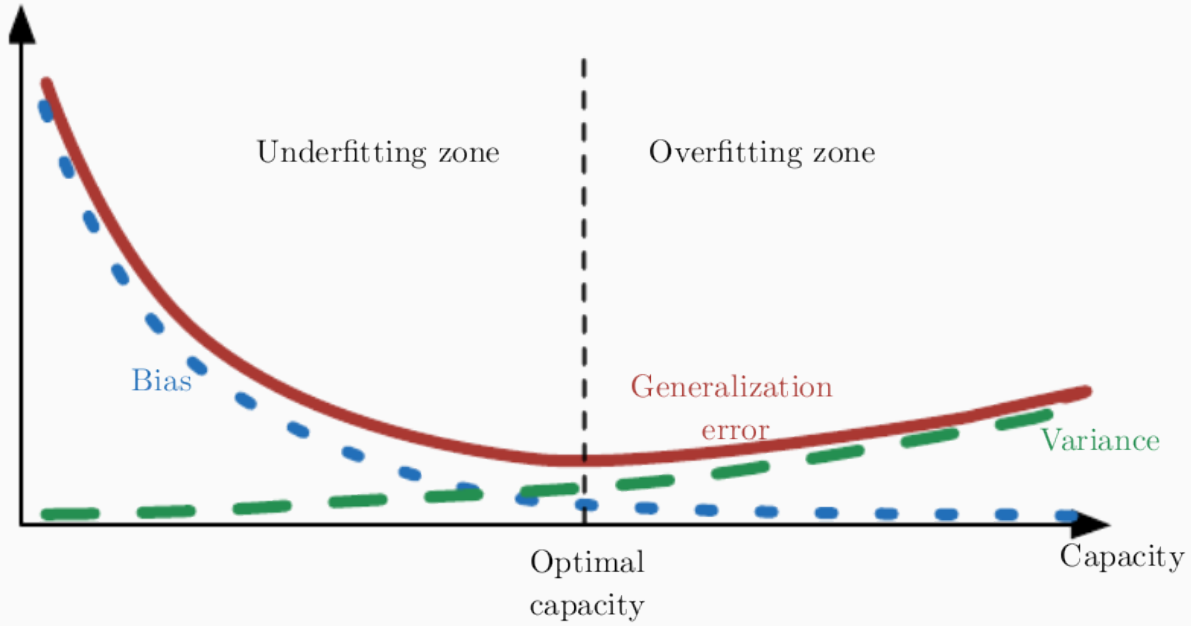
So, the expectation averages over the variability of y_0 (bias) and the variability in the training data.

Bias-variance trade-off

If $\hat{y}$ increases **flexibility**, its variance increases and its biases decreases.

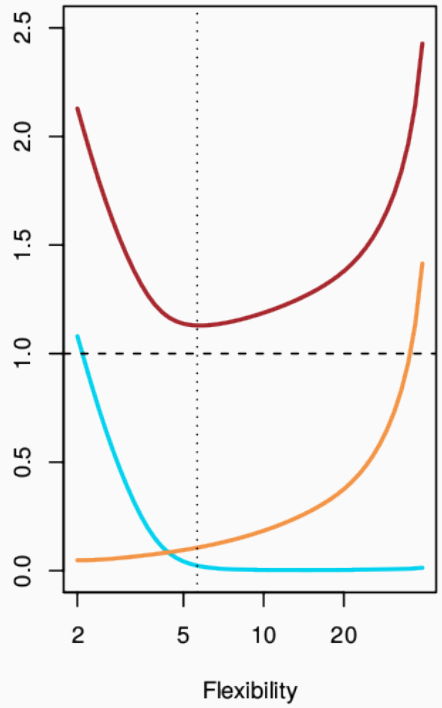
Choosing the flexibility based on average test error amounts to a bias-variance trade-off:

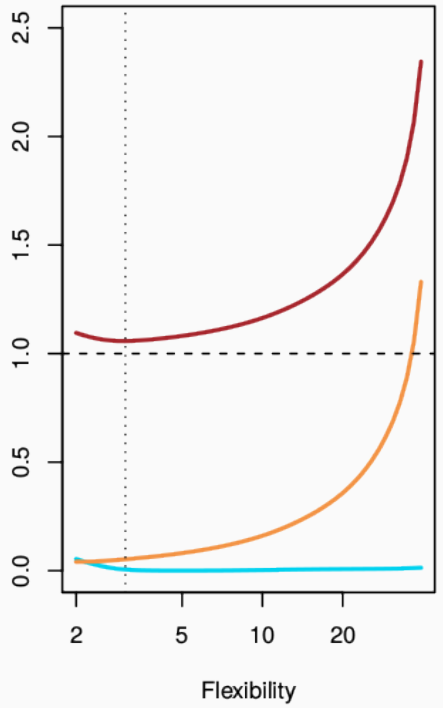
- **High Bias** $\rightarrow$ underfitting:
  erroneous assumptions in the learning algorithm.
- **High Variance** $\rightarrow$ overfitting:
  erroneous sensitivity to small fluctuations (noise) in the training set.

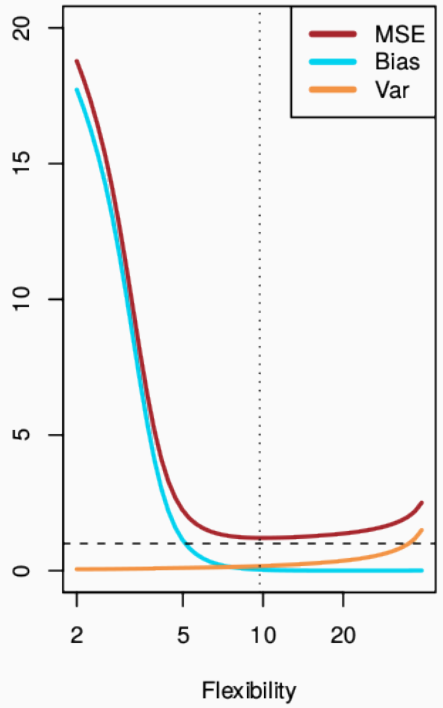


Bias-variance trade-off

More examples of bias-variance trade-off:







Bias-variance trade off

Regularization techniques can be applied to modify the learning algorithm and reduce its generalization error but not its training error.

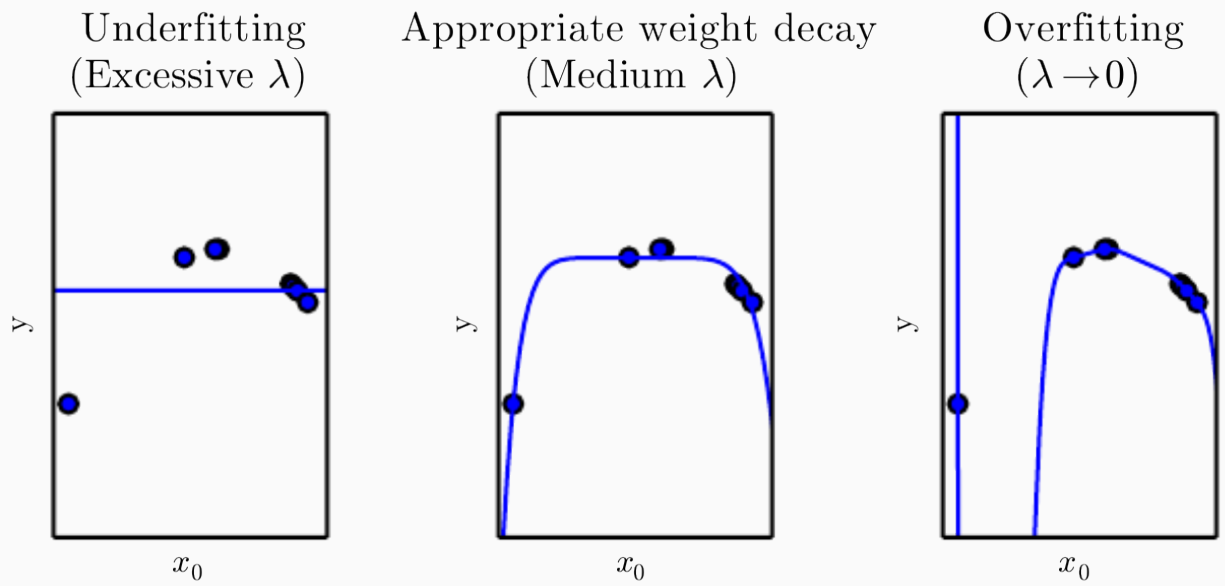
For example, including the weight decay to the MSE cost function:

$$J(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_{\boldsymbol{w}}(\boldsymbol{x}_i))^2 + \lambda \boldsymbol{w}^T \boldsymbol{w}.$$

where λ is a real number which express the preference for weights with smaller squared L^2 norm.

Solution for the bias-variance trade off

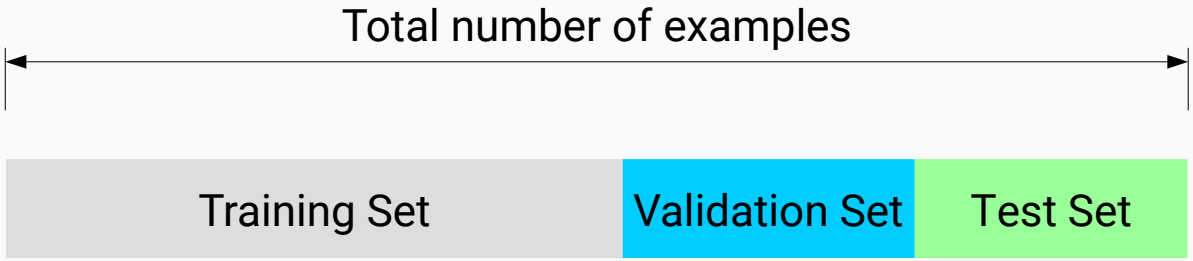
Tuning the hyperparameter λ we can regularize a model without modifying explicitly its capacity.



Solution for the bias-variance trade off

A common way to reduce the bias-variance trade-off and choose the proper learning hyperparamters is to create a validation set that:

- not used by the training algorithm
- not used as test set



- **Training set**: examples used for learning.
- **Validation set**: examples used to tune the hyperparameters.
- **Test set**: examples used only to access the performance.

Techniques are available to deal with data samples with large and small number of examples. (talk later)

Assessing model performance for classification

In binary classification tasks we usually complement the cost function with the accuracy metric defined as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}.$$

Example:

True Positives (TP) e.g. 8	False Positives (FP) e.g. 2
False Negatives (FN) e.g. 4	True Negatives (TN) e.g. 20

- Accuracy $= 82\%$

Assessing model performance for classification

In binary classification tasks we usually complement the cost function with the accuracy metric defined as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}.$$

Example:

True Positives (TP) e.g. 8	False Positives (FP) e.g. 2
False Negatives (FN) e.g. 4	True Negatives (TN) e.g. 20

- Accuracy = 82%

However accuracy does not represents the overall situation for skewed classes, *i.e.* imbalance data set with large disparity, *e.g.* signal and background.

In this cases we define precision and recall.

Assessing model performance for classification

Precision: proportion of correct positive identifications.

Recall: proportion of correct actual positives identifications.

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}$$

True Positives (TP) e.g. 8	False Positives (FP) e.g. 2
False Negatives (FN) e.g. 4	True Negatives (TN) e.g. 20

- Accuracy $= 82\%$
- Precision $= 80\%$
- Recall $= 67\%$

Assessing model performance for classification

Precision: proportion of correct positive identifications.

Recall: proportion of correct actual positives identifications.

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}$$

True Positives (TP) e.g. 8	False Positives (FP) e.g. 2
False Negatives (FN) e.g. 4	True Negatives (TN) e.g. 20

- Accuracy $= 82\%$
- Precision $= 80\%$
- Recall $= 67\%$

Various metrics have been developed that rely on both precision and recall, e.g. the F_1 score:

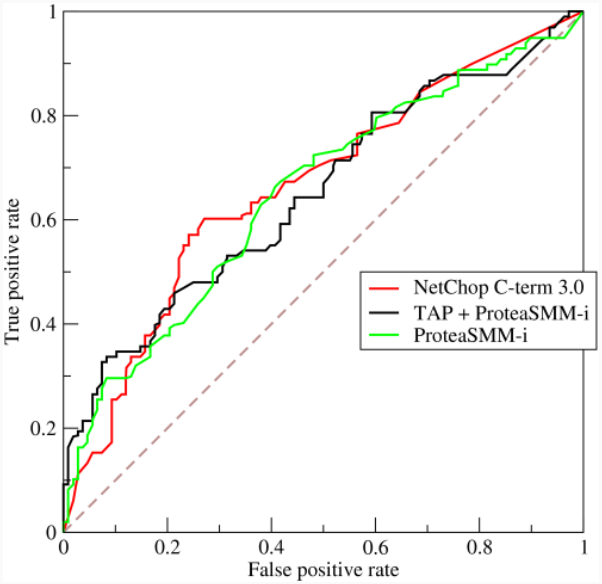
$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = 73\%$$

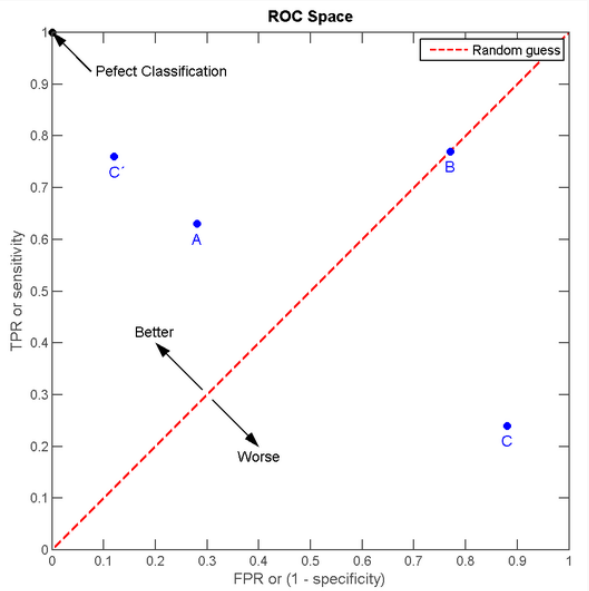
Assessing model performance for classification

In a binary classification we can vary the probability threshold and define:

- the **receiver operating characteristic** curve (ROC curve) is a metric which shows the relationship between correctly classified positive cases, the true positive rate (TRP/recall) and the incorrectly classified negative cases, false positive rate (FPR, (1-effectivity)).

$$\mathrm{TPR} = \frac{TP}{TP + FN}, \quad \mathrm{FPR} = \frac{FP}{FP + TN}$$



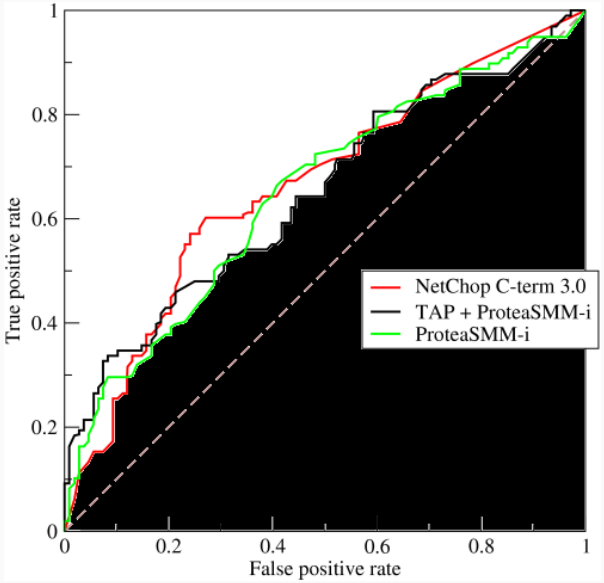


Assessing model performance for classification

The **area under the ROC curve** (AUC) represents the probability that classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one.

AUC provides an aggregate measure of performance across all possible classification thresholds.

- AUC is 0 if predictions are 100% wrong
- AUC is 1 if all predictions are correct.
- AUC is scale-invariant and classification-threshold-invariant.



Summary

Summary

We have covered the following topics:

- Motivation and overview of A.I.
- Definition and overview of ML.
- Model representation definition and trade-offs
- Learning metrics for accessing the model performance
- Metrics for classification.