Machine Learning Lecture

Ecole Internationale de Physique Subatomique
IPNL, Lyon
22-26 October 2018

Jean-Roch Vlimant
California Institute of Technology

Focus of the Lectures

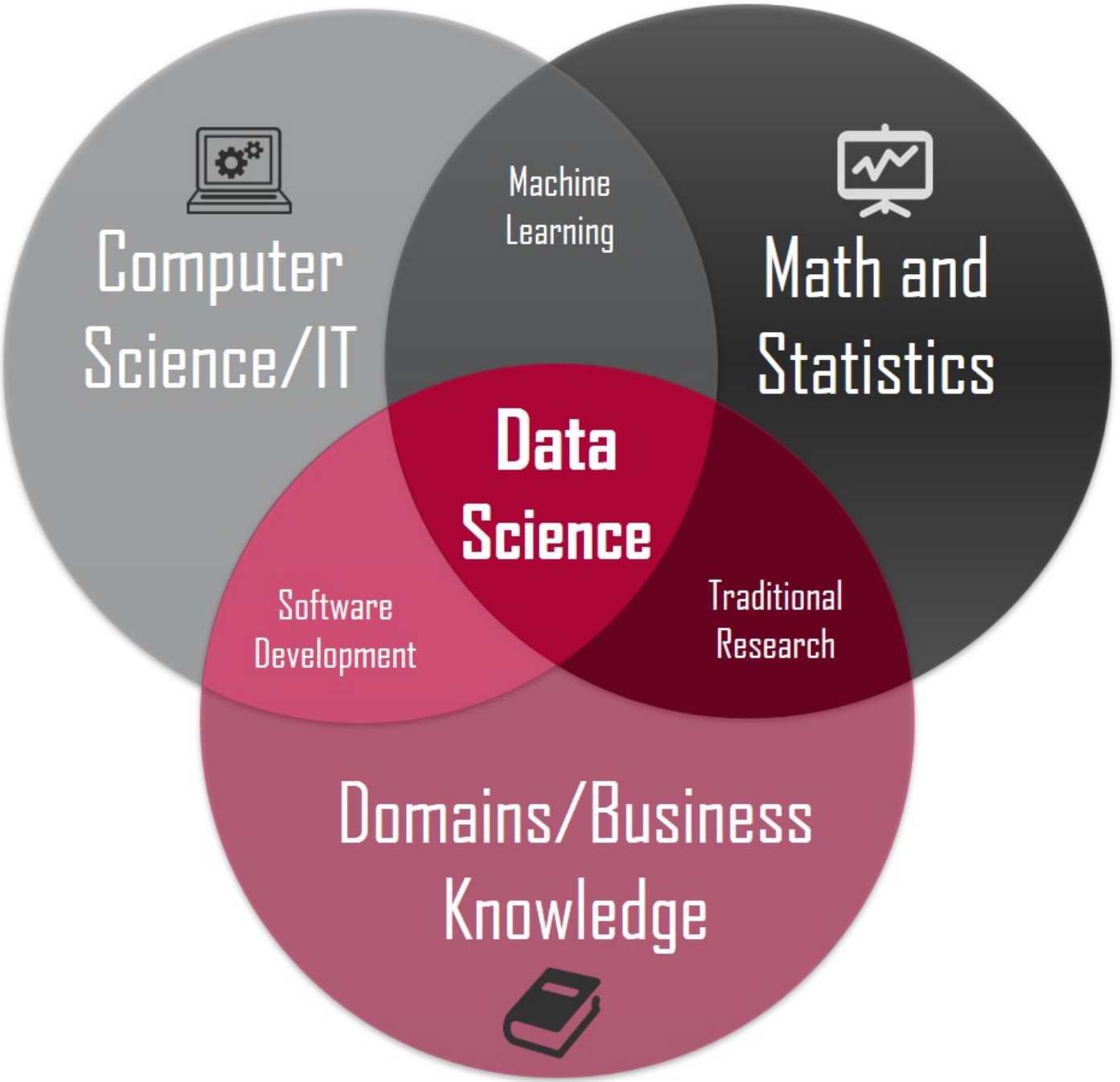
- Provide enough details to understand the big picture of machine learning and some methods
- Leaving further understanding and computational details for the hands-on sessions and personal work
- Provide an overview of what is “out there” for you to go and do more research for your particular problem/application
- Inspiration taken from previous such lectures (see references slide)
- Cannot cover what would deserve a full series of computer science lectures

Outline

- Day 1
  - ➔ Lecture :
    - Generalities

- Day 2
  - ➔ Lecture :
    - Unsupervised learning
    - Supervised learning
      - Artificial Neural Network
  - ➔ Hands-on
    - Dataset manipulation
    - Scikit-learn
    - keras, tensorflow, pytorch

- Day 3
  - ➔ Lecture
    - Cutting Edge technique


Computer Science/IT
Machine Learning
Math and Statistics
Data Science
Software Development
Traditional Research
Domains/Business Knowledge

Lecture - Part 1/3

Generalities : Outline

- Overview
- Optimization Methods
- Model Uncertainty
- Model Balance
- Assembling Models
- Training & Inference

What Is Machine Learning

"Giving computers the ability to learn without explicitly programming them" A. Samuel (1959).

Is fitting a straight line machine learning ?
Models that have enough capacity to define its own internal representation of the data to accomplish a task : **learning from data.**

In practice : a statistical method that can extract information from the data, not obviously apparent to an observer.

- Most approach will involve a **mathematical model** and a cost/reward function that needs to be **optimized.**
- The more **domain knowledge** is incorporated, the better.

Overview

Reinforcement Learning (cherry)

- The machine predicts a scalar reward given once in a while.
- **A few bits for some samples**

Supervised Learning (icing)

- The machine predicts a category or a few numbers for each input
- **10→10,000 bits per sample**

Unsupervised Learning (cake)

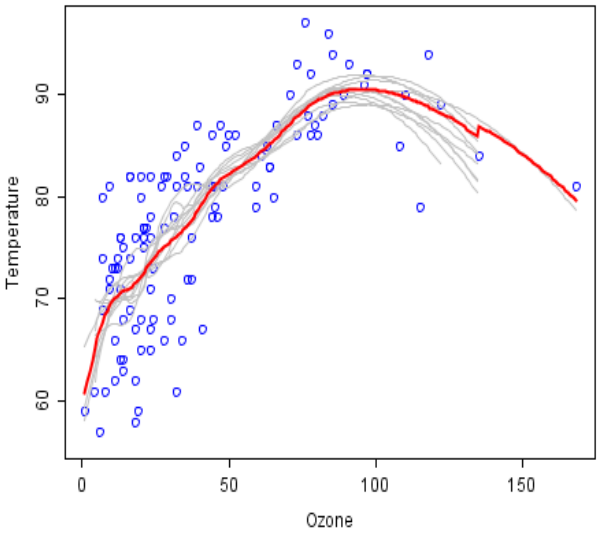
- The machine predicts any part of its input for any observed part.
- Predicts future frames in videos
- **Millions of bits per sample**

Yann Le cun, CERN 2016

Supervised Learning

- Given a dataset of samples, a subset of features is qualified as **target**, and the rest as **input**
- Find a **mapping from input to target**
- The mapping should **generalize to any extension** of the given dataset, provided it is generated from the same mechanism

$$dataset \equiv \left\{ \left(x_i , y_i \right) \right\}_i$$
$$find \;\; function \;\; f \;\; s.t. \;\; f\left(x_i \right) = y_i$$



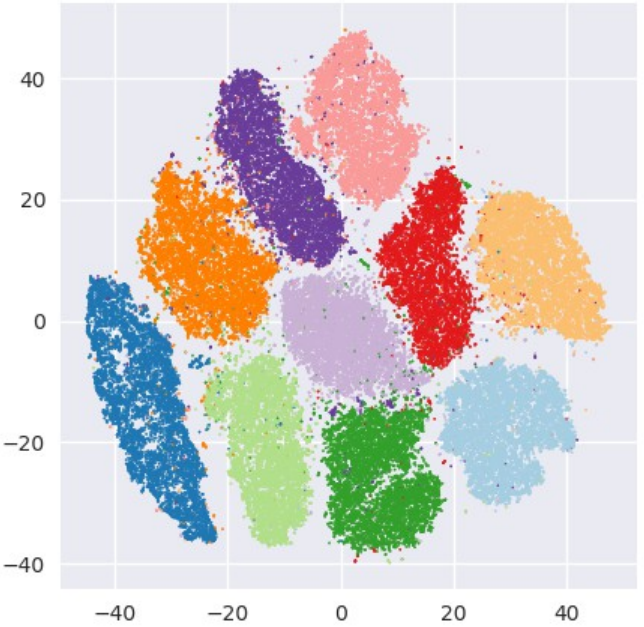
- Finite set of target values :
  - ➔ **Classification**
- Target is a continuous variable :
  - ➔ **Regression**

Unsupervised Learning

- Given a dataset of samples, but there is no subset of feature that one would like to predict
- Find mapping of the samples to a lower dimension manifold
- The mapping should generalize to any extension of the given dataset, provided it is generated from the same mechanism

$$dataset \equiv \{(x_i)\}_i$$
$$find \;\; f \;\; s.t. \;\; f(x_i) = p_i$$

- Manifold is a finite set
  - → **Clusterization**
- Manifold is a lower dimension manifold :
  - → **Dimensionality reduction, density estimator**



Reinforcement Learning

- Given an **environment** with multiple states, given a reward upon action being taken over a state
- Find an **action policy to drive** the environment toward maximum cumulative reward

$$s_{t+1} = Env(s_t, a_t)$$
$$r_t = Rew(s_t, a_t)$$
$$\pi(a|s) = P(A_t = a | S_t = s)$$
$$find\ \pi\ s.t.\ \sum_t r_t\ is\ maximum$$

Motivation

Classical (read not deep-learning) machine learning has been around for long and **used at many level** in science.

Artificial neural network : a.k.a “Deep learning” is now very present in data-science thanks to :

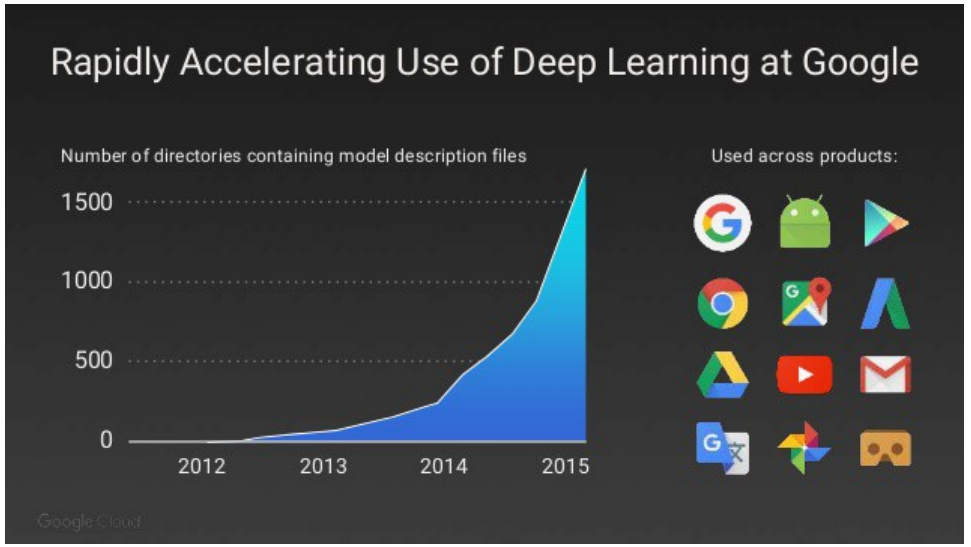
- Increased **computation power** through general purpose graphical processing units (GP-GPU)
- Increased **dataset size** through the internet-of-things (IOT)
- Improved **models architectures** (relu activation, convolution, …)

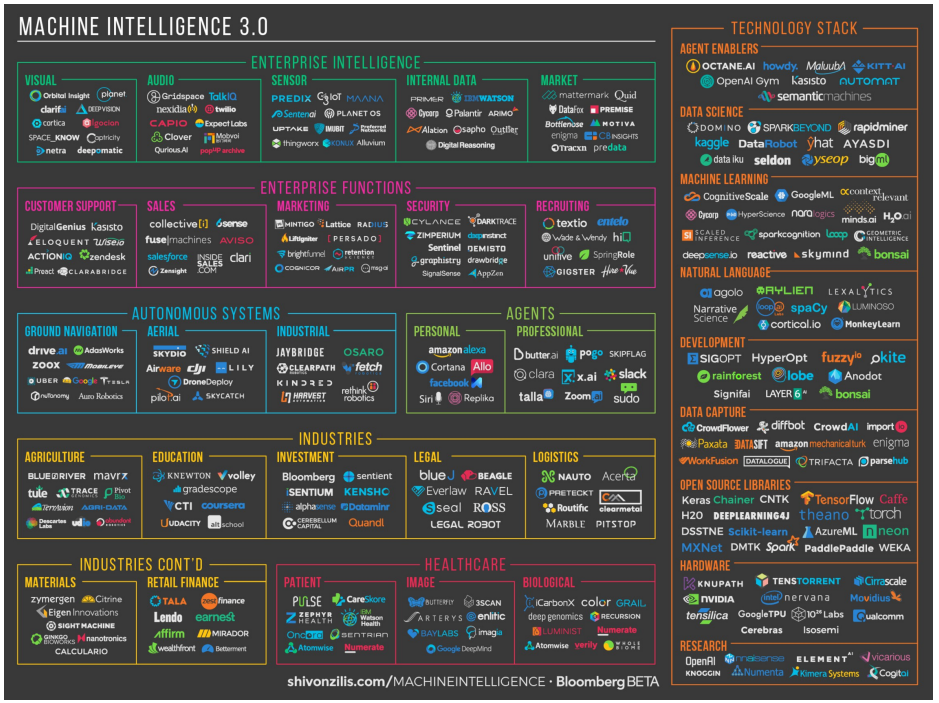
→ It became possible to **train models with millions of parameters** on dataset with **millions of samples**, each with **multiple thousands of pixels**

→ It became possible to **extract very complex correlations**, otherwise cumbersome to model.

Machine Learning in Industry

https://www.nvidia.com/en-us/deep-learning-ai/





http://www.shivonzilis.com/machineintelligence

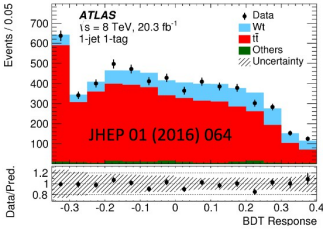
- Prominent field in industry nowadays
- Lots of data, lots of applications, lots of potential use cases, lots of money

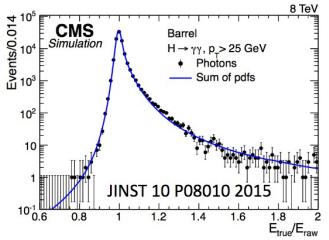
➔ Knowing machine learning can open significantly your **career horizons**

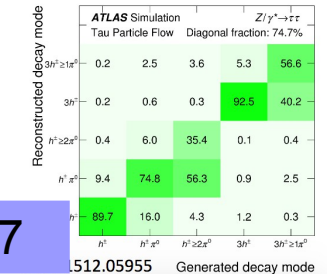
Machine Learning in HEP

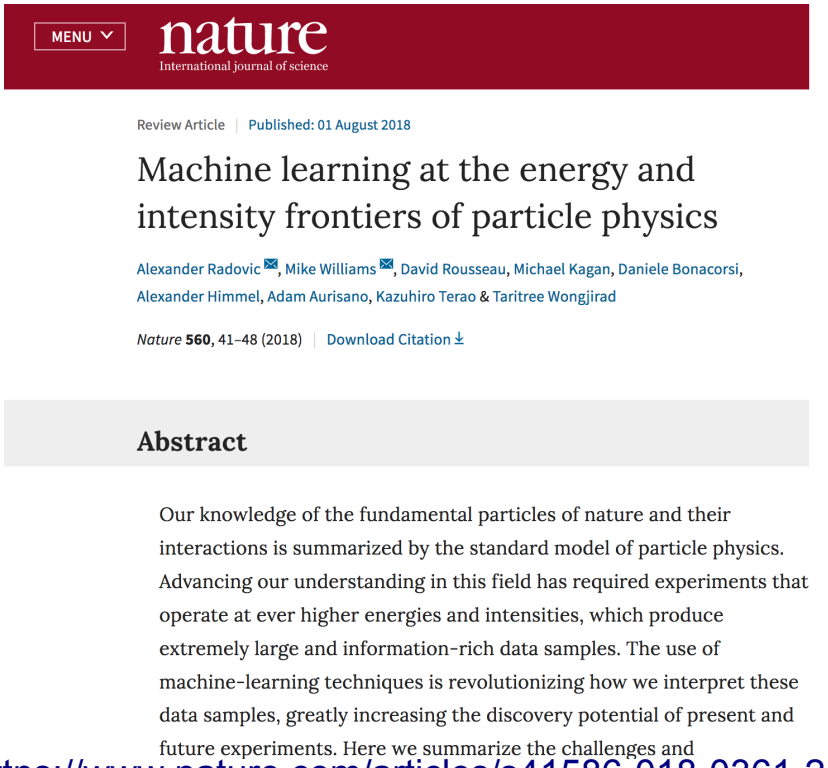
- **In analysis**:
  - Classifying signal from background, especially in complex final states
  - Reconstructing heavy particles and improving the energy / mass resolution
  - …
- **In reconstruction**:
  - Improving detector level inputs to reconstruction
  - Particle identification tasks
  - Energy / direction calibration
  - …
- **In the trigger**:
  - Quickly identifying complex final states
  - …
- **In computing**:
  - Estimating dataset popularity, and determining how number and location of dataset replicas
  - …

Michael Kagan, CERN 2017





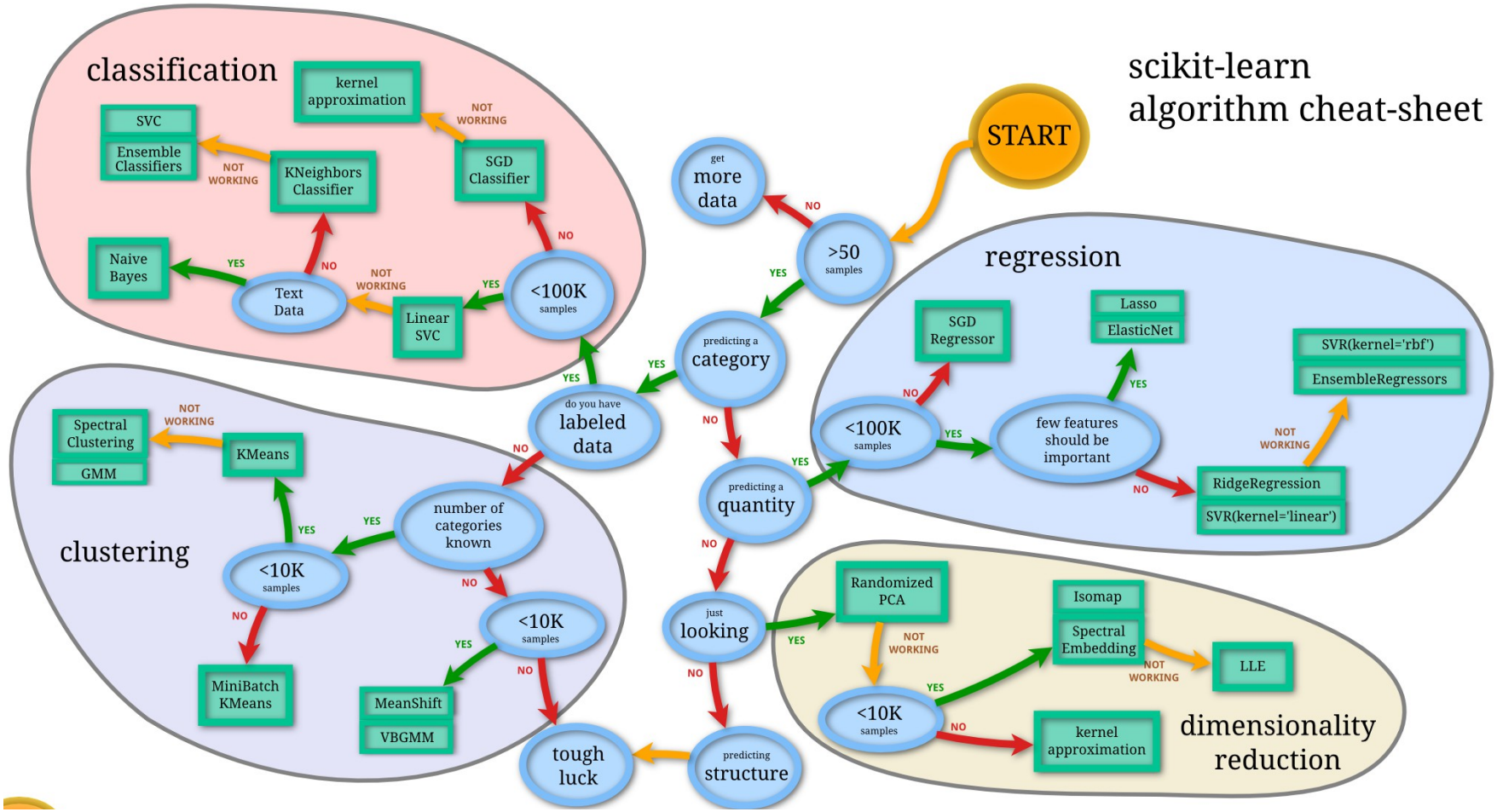




https://www.nature.com/articles/s41586-018-0361-2

- Machine Learning and Deep learning brought significant improvement to the field
- Needs to be used with care and scientific approach

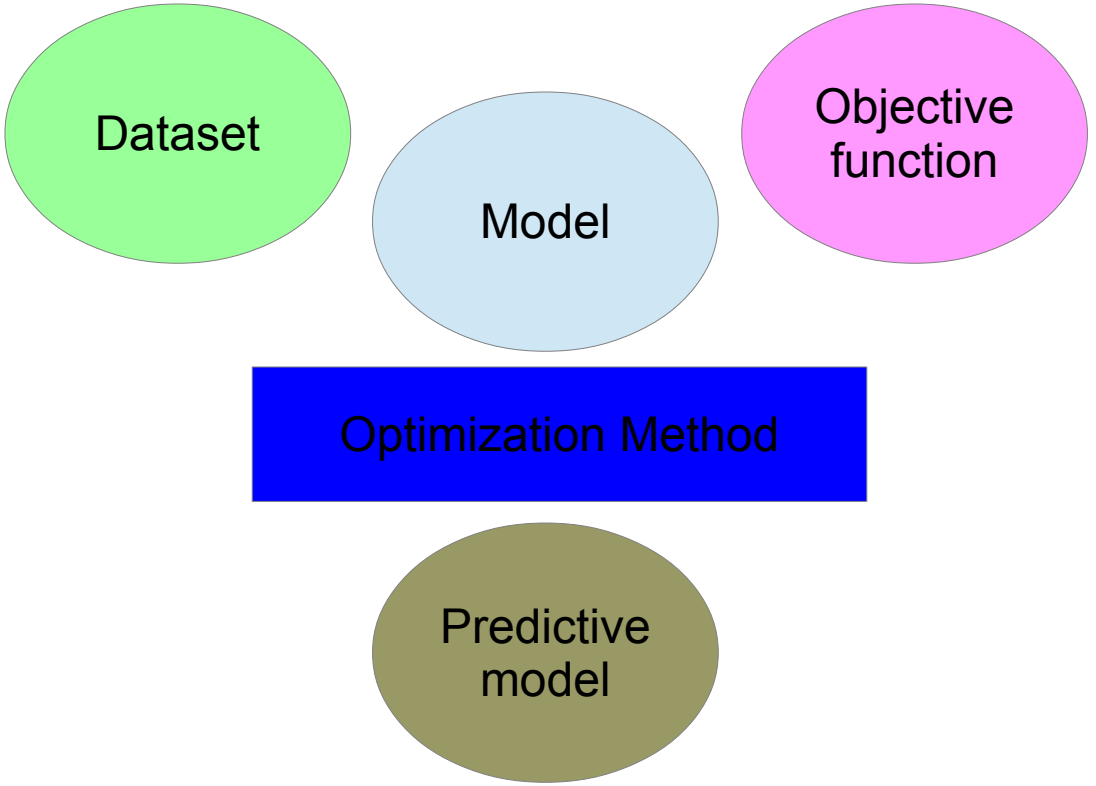
(Some) Machine Learning Methods



http://scikit-learn.org/stable/tutorial/index.html

- There are a lot of methods out there

Overview



- Many optimization methods adapted to the various phase space of the dataset, model, objective
- Gradient descent, evolutionary algorithms, ...

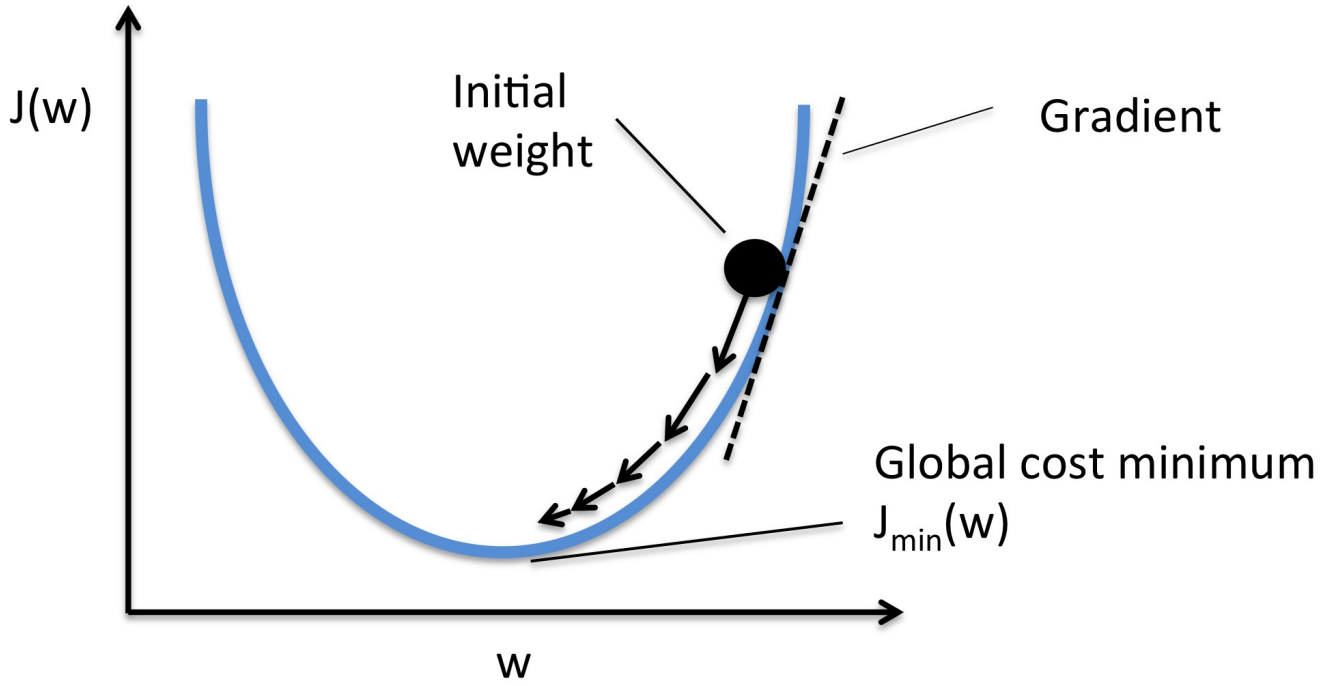
Objective Function

- Named Objective/Loss/Cost/Utility/Reward function interchangeably
- Represents the function that represents the goodness of a model at solving the problem at hand
- Function to be optimized to find the best model

- e.g. Mean squared difference for regression

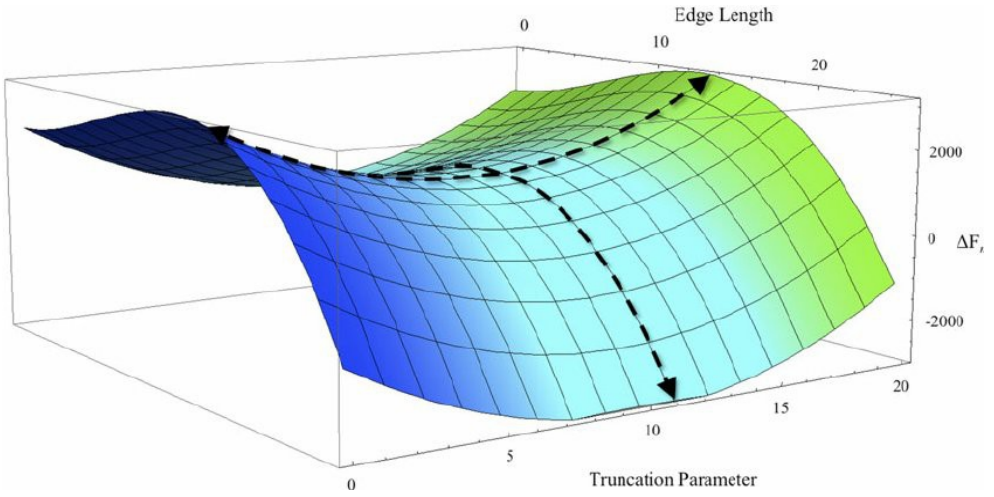
Optimization Methods

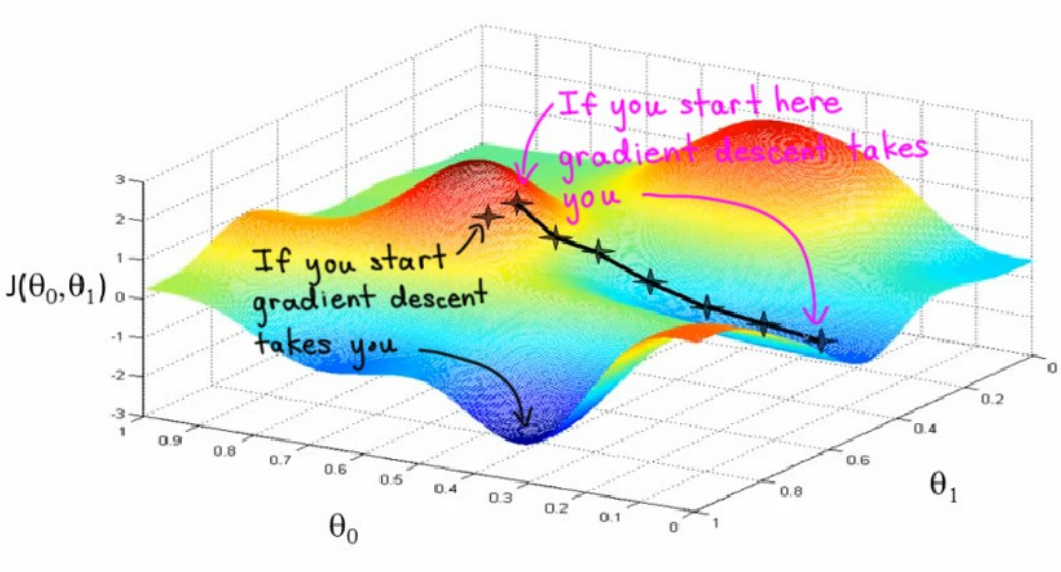
Gradient Descent Optimization



- For a differentiable loss function f, the first Taylor expansion gives $f(x+\varepsilon)=f(x)+\varepsilon\nabla f(x)$
- The direction to locally maximally decrease the function value is anti-collinear to the gradient $\varepsilon=-\gamma\nabla f(x)$
- Amplitude of the step γ to be taken with care to prevent overshooting

Non-Convex Optimization





- The objective functions optimized in machine learning are usually non-convex
- Non guaranteed convergence of gradient descent
- Gradients may vanish near local optimum and saddle point

Stochastic Gradient Descent

- Application of one gradient descent is expensive. Can be prohibitive with large datasets
- Following the gradient update from each and every sample of a dataset leads to tensions
    - In binary classification, samples from opposite categories would have “opposite gradients”
- Gradients over multiple samples are independent, and can be computationally parallelyzed

➔ Estimate the effective gradient over a batch of samples

$$\nabla_{eff} f(x) = \frac{1}{N} \sum_{i \in batch} \nabla_i f(x)$$

Non Analytical SGD

- Some valuable loss function might not be analytical and their gradients cannot be derived
- Used finite element method to estimate the gradient numerically

$$\nabla f(x) = \frac{f(x+\varepsilon) - f(x)}{\varepsilon}$$

- Method can be extended to using more sampling and better precision
- Quite expensive computationally in number of function calls and impractical in large dimension
- Robust methods available in most program library

Second Order Methods

- Newton-Raphson method defines a recursive procedure to find the root of a function, using its gradient.
- Finding optimum is equivalent to finding roots of the gradient, hence applying NR method to the gradient using the Hessian

$$f(x+\varepsilon) = f(x) + \varepsilon \nabla f(x) + \frac{1}{2}\varepsilon^T H(x)\varepsilon$$

$$\varepsilon \sim -H(x)^{-1}\nabla f(x)$$

- Convergence guaranteed in certain conditions
- Alternative numerical methods tackle the escape of saddle points and computation issue with inverting the Hessian
- In deep learning “hessian-free” methods are prohibitive computationally wise

Approximate Bayesian Computation

$$\pi(model|data) = \frac{\pi(data|model)\,\pi(model)}{\pi(data)}$$

- ABC is applicable when the likelihood $\pi(data|model)$ is intractable/unknown
- The method requires a simulator or surrogate model
- Generate simulated data for models drawn from the prior, accept/reject whether matching data

- Overly expensive in calls to simulator
  - Introduce summary statistics to enhance border cases
  - Efficient sampling to boost acceptable models
  - Generalized methods for comparing simulated samples with data

➔ Most relevant work on likelihood-free inference in HEP https://arxiv.org/abs/1805.12244

Bayesian Optimization

- Applicable to optimize function **without close form** and that are **expensive to call** (numerical gradient impractical)
- Approximate the objective function with **Gaussian processes** (GP)
- Start at random points, then sample according to optimized acquisition function
  - Expected improvement

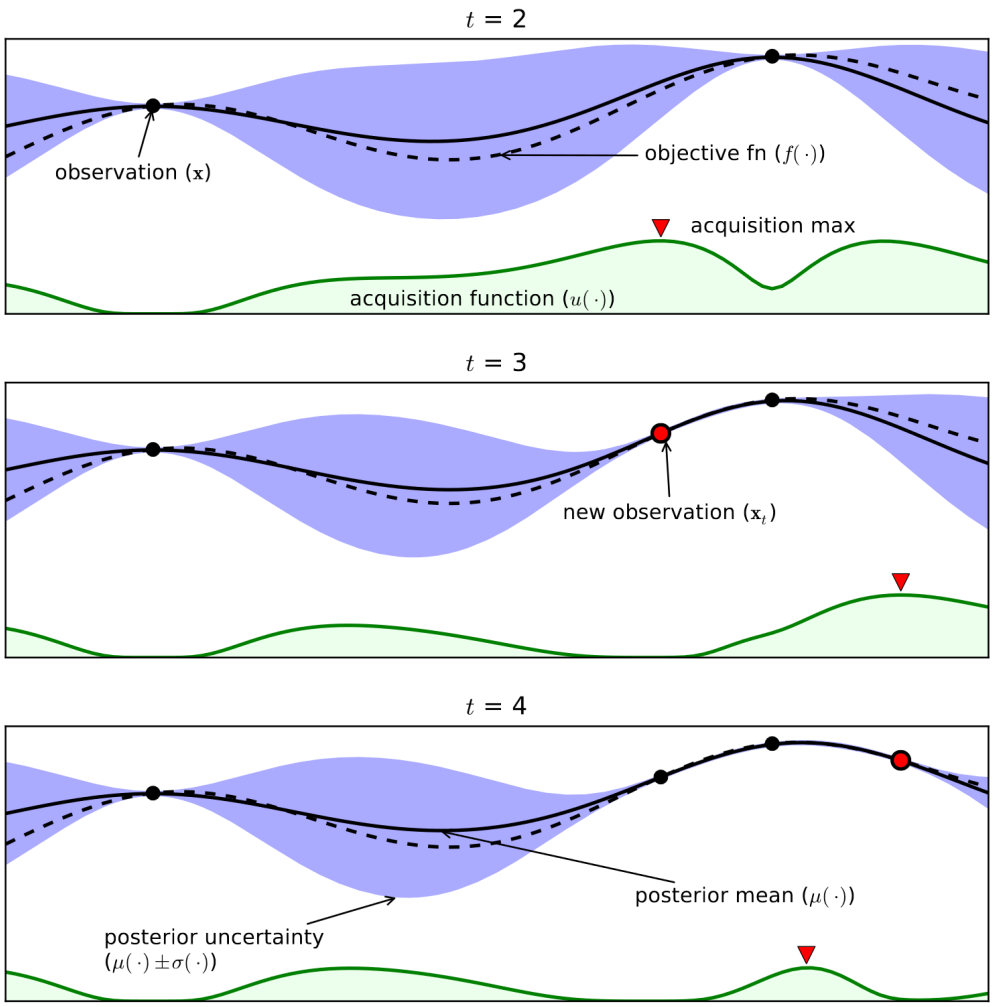
$$-EI(x) = -E(f_{GP}(x) - f(x_{best}))$$

  - Lower confidence bound

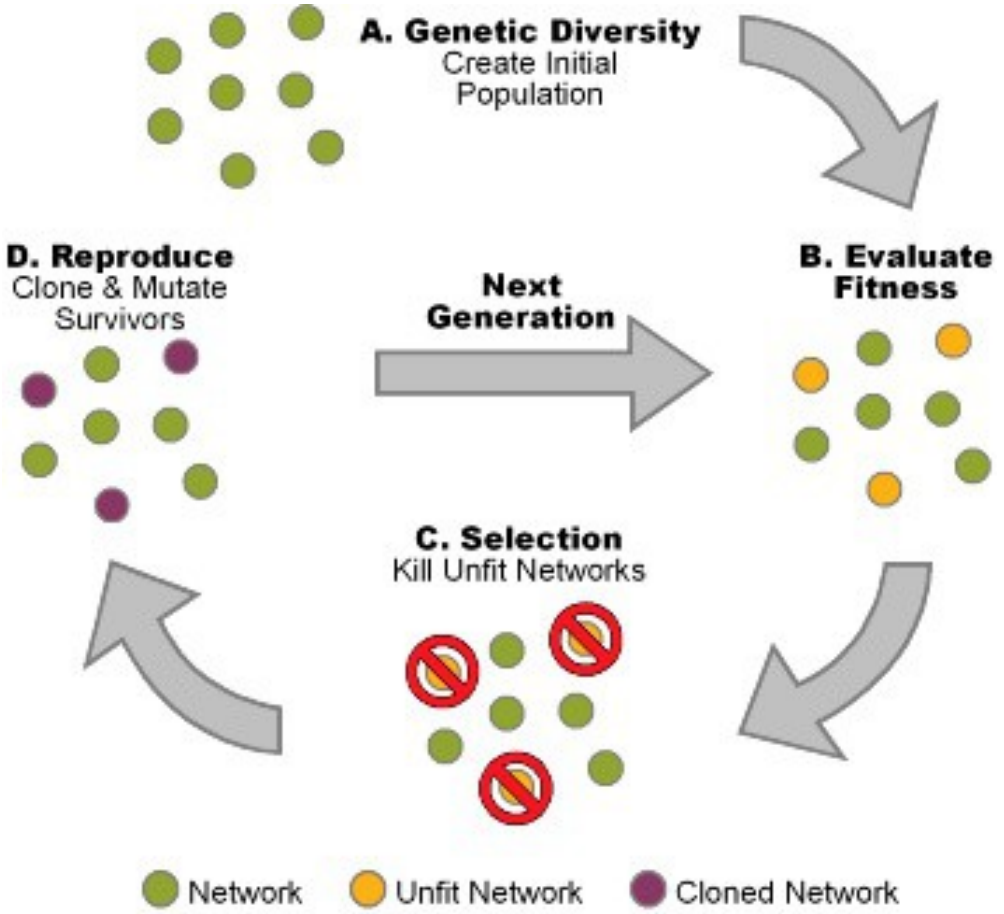
$$LCB(x) = \mu_{GP}(x) + \kappa\,\sigma_{GP}(x)$$

  - Probability of improvement

$$-PI(x) = -P(f_{GP}(x) \geq f(x_{best}) + \kappa)$$



Evolutionary Algorithms

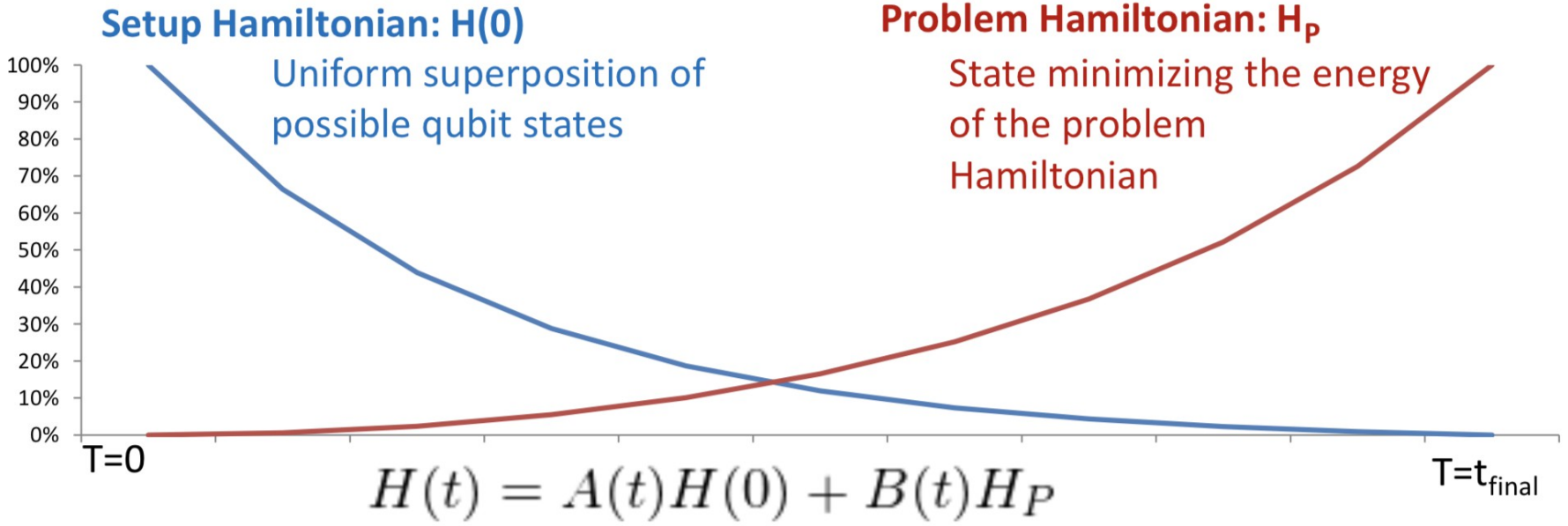


- Applicable to function in high dimensions, with a non regular landscape
- Start from random population
- Estimate fittest fraction of individuals
- Bread and mutate individuals

- Direction of optimization is given by the cross-over and mutation definition
- Multiple over algorithms : particle swarn, ...

Adiabatic Quantum Annealing

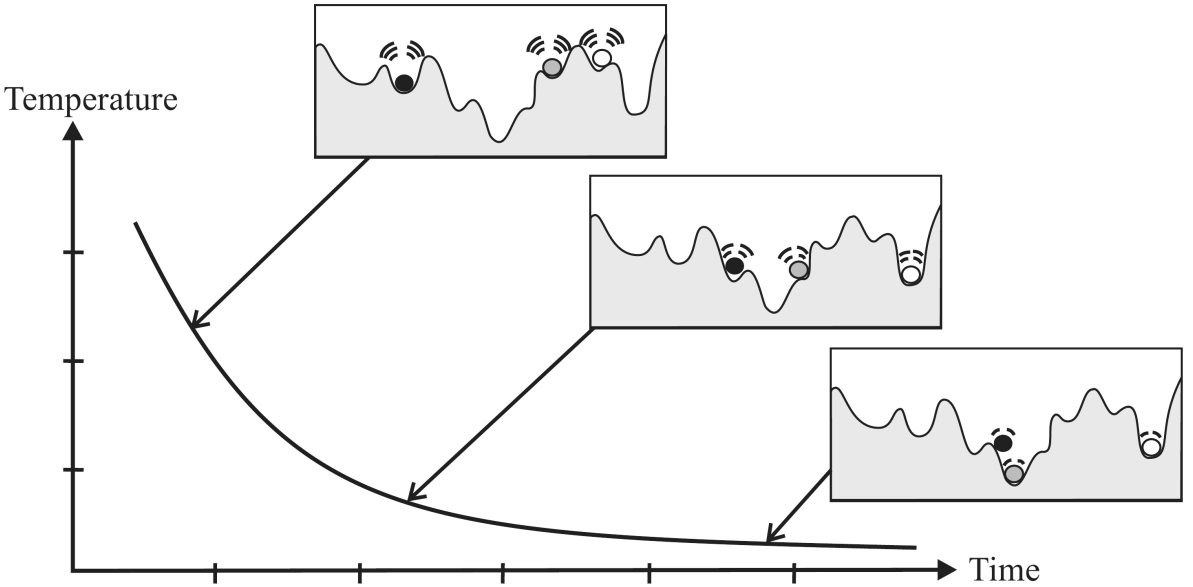
- System setup with trivial Hamiltonian H(0) and ground state
- Evolve adiabatically the Hamiltonian towards the desired Hamiltonian H_p
- **Adiabatic theorem** : with a slow evolution of the system, the state stays in the ground state.



https://arxiv.org/abs/quant-ph/0001106
https://arxiv.org/abs/quant-ph/0104129

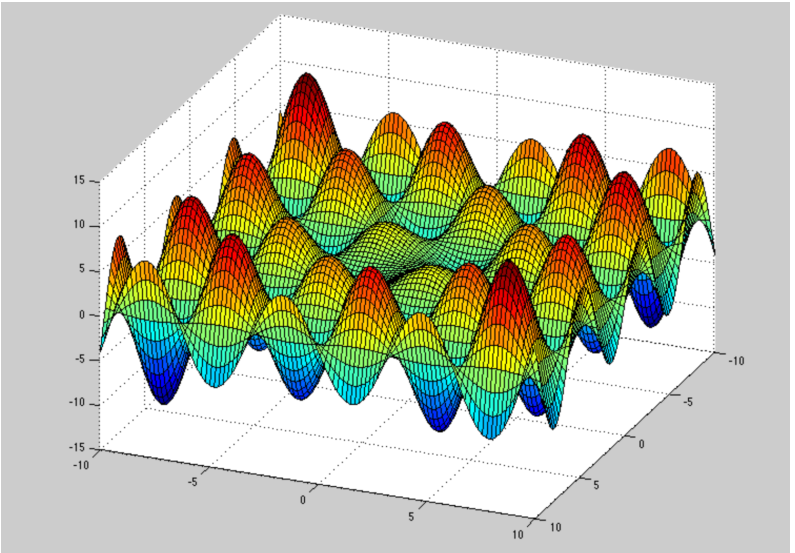
Simulated Annealing

- Monte-Carlo based method to find ground state of energy functions
- Random walk across phase space
  - ➔ accepting descent
  - ➔ accepting ascent with probability $e^{-\Delta E/kT}$
- Decrease T with time




Model Uncertainty

Initialization



- chaos ≡ small change of initial condition can lead to large difference in output
- Optimization of non-convex function is chaotic
  - Degenerate local/global minimums
  - Length scale of stepping in optimization
- Choice of initial point(s) in parameter space influences the outcome of the optimization
- Needs to be accounted for in model comparison
- Quoted performance should include an estimation of the variance

Test Statistics

- Model performance estimated on a finite sized dataset
- Can alter the outcome of model comparison
- Particularly important in dealing with small number of samples per class
- Needs to be properly taken into consideration during model comparison
- Model performance should be reported with an estimation of that uncertainty

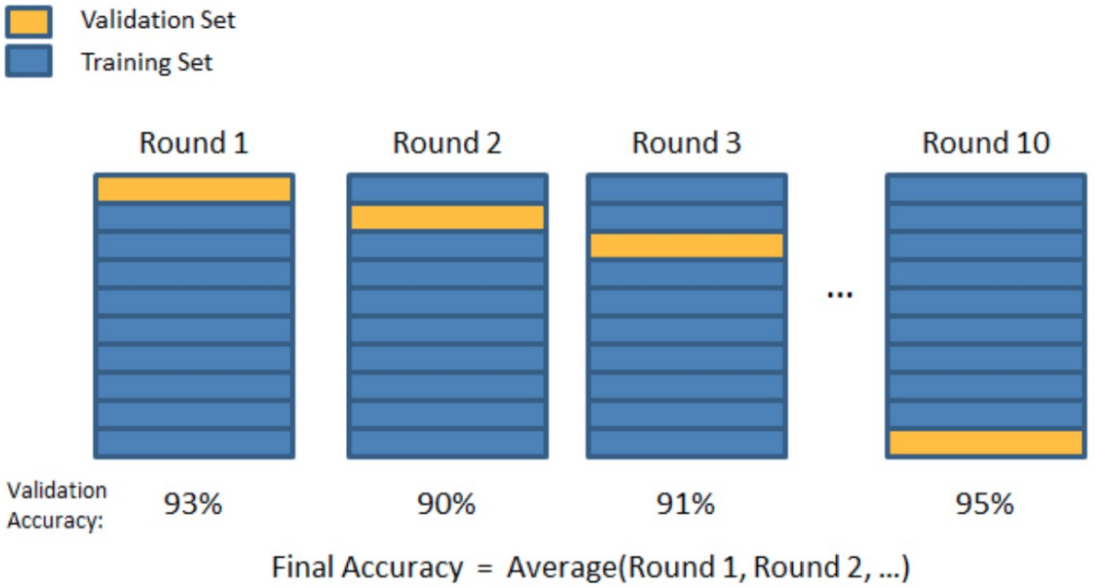
- Can be estimated with bootstrapping for example

$$\{p_i\}_{i \in test\ set} \sim Poisson(1)$$

$$fom^* = \underset{\{p_i\}}{E}\left[fom(x_i\, p_i)\right]$$

$$\sigma^*_{fom} = \underset{\{p_i\}}{RMS}\left[fom(x_i\, p_i)\right]$$

Cross Validation

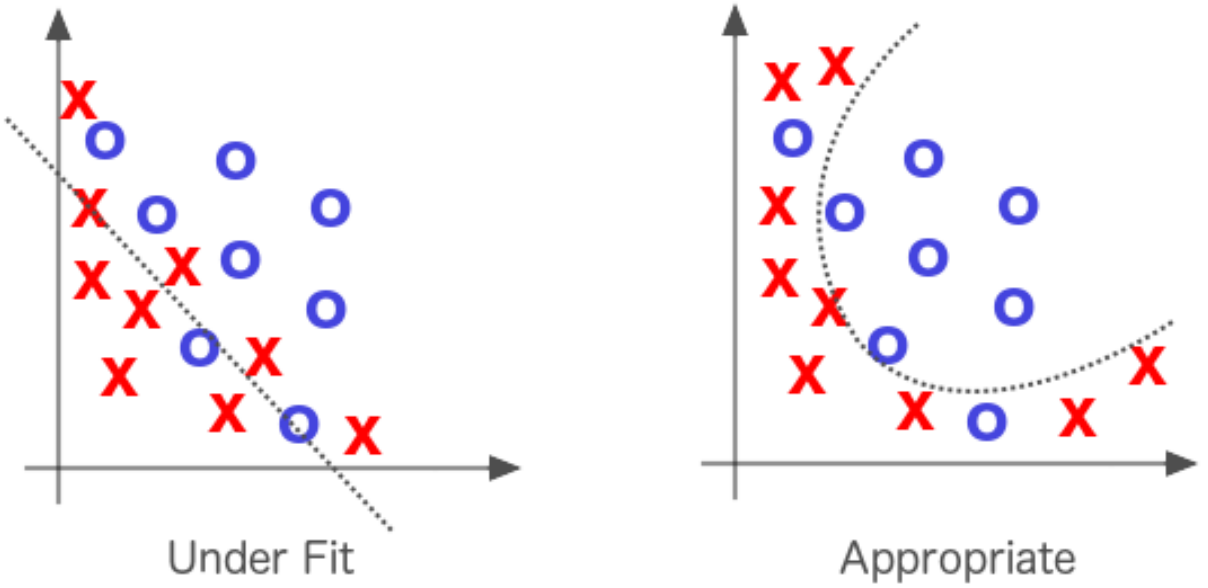


- Model selection requires to have an estimate of the uncertainty on the metric used for comparison
- K-folding provides an un-biased way of comparing models
- Stratified splitting (conserving category fractions) protects from large variance coming from biased training
- Leave-one-out cross validation : number folds ≡ sample size


Model Balance

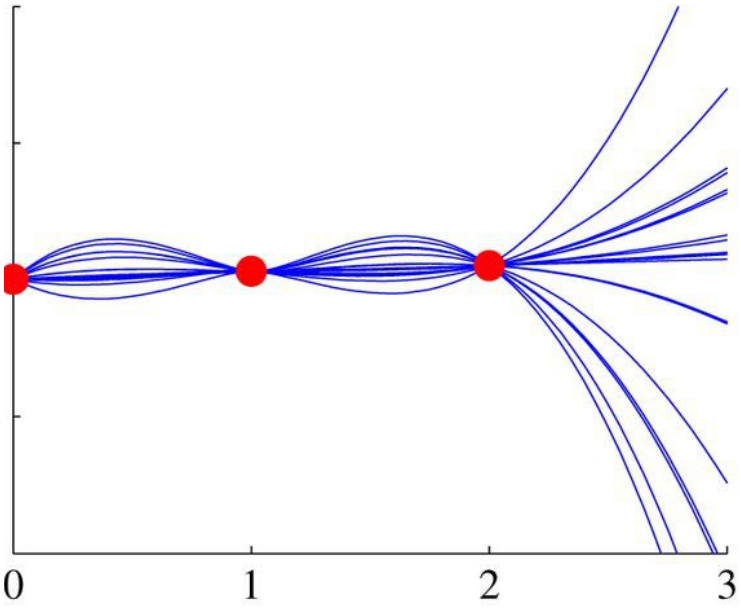
Under-fitting

- Poor model performance can be explained
  - Lack of modeling capacity (not enough parameters, inappropriate parametrization, …)
  - Model parameters have not reached optimal values



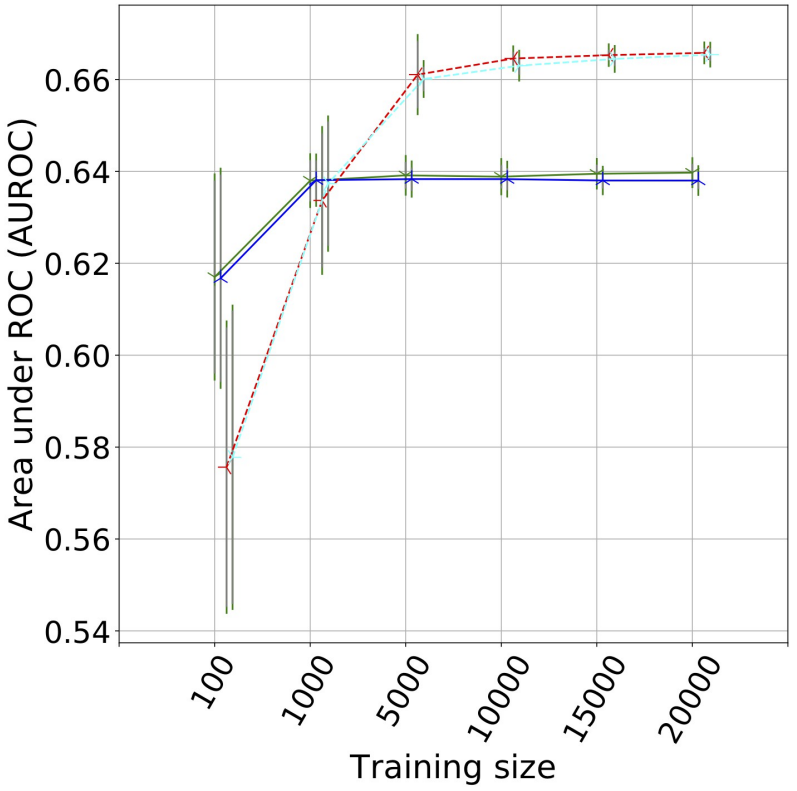
Data Efficiency

- Degeneracy in objective function can arise from under-constrained model
  - Not enough data point to constraint all parameters
- Family of solutions make the optimization algorithm stationary on the training set, while fluctuating on the testing set
- Limit of under-constraint is model type dependent.
- Can usually happen with deep learning models



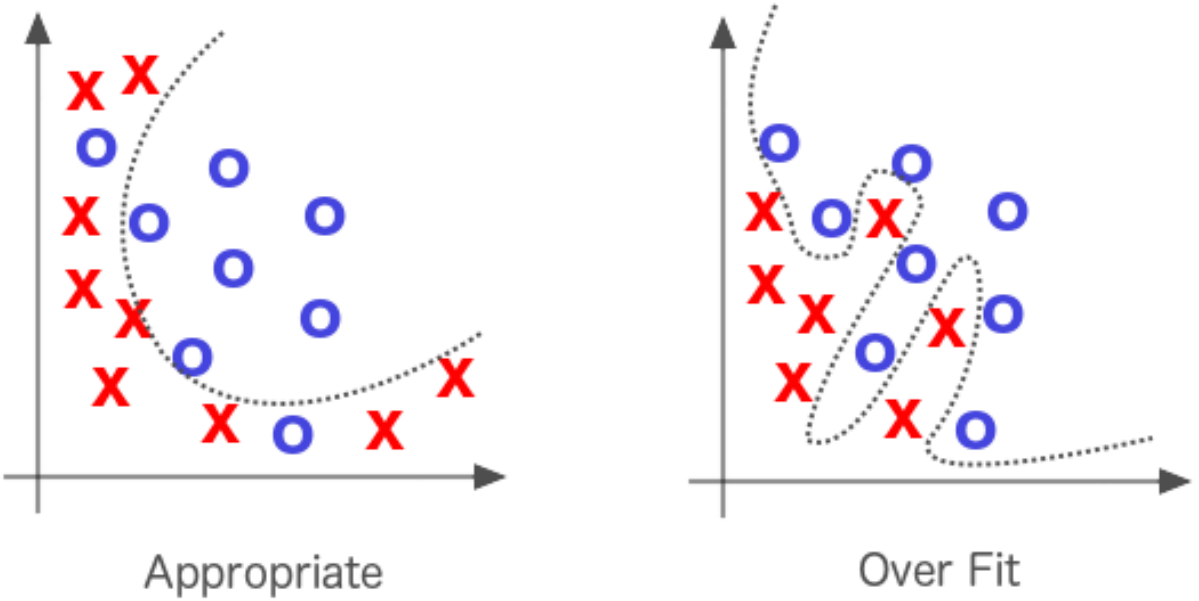
Need for Data

- "What is the **best performance one can get** ?" rarely has an answer
- When comparing multiple models, one can answer "what is the **best of these models, for this given dataset** ?"
- It does not answer "what is the **best model at this task** ?"



Over-fitting

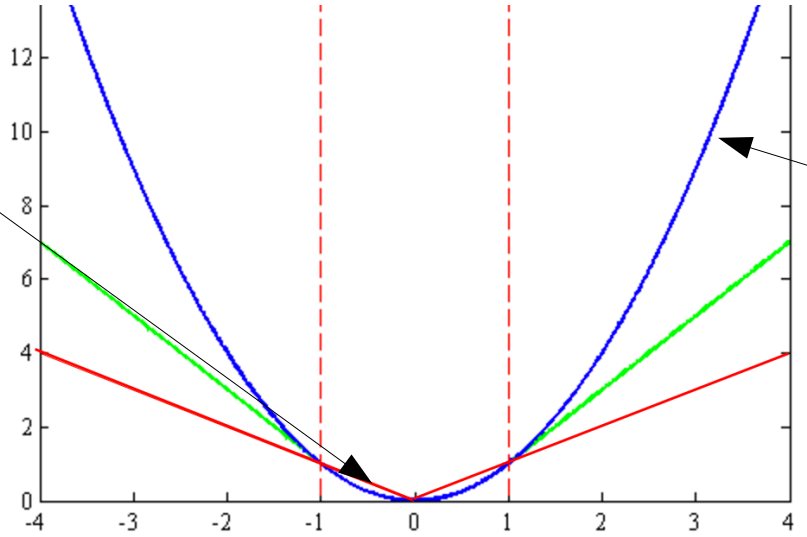
- "Too good to be true" model performance can be explained
  - Excessive modeling capacity (too many parameters, parametrization is too flexible, ...)
  - Model parameters have learn the trained data by heart
- Characterized by very good performance on the training set and (much) lower performance on unseen dataset



Regularization

- Over-fitting can be limited by additional terms to the objective function
- Any term that aim at reducing the capacity of the model
  - L1 : $Loss_{reg} = Loss + \lambda_1 \sum |w_i|$
  - L2 : $Loss_{reg} = Loss + \lambda_2 \sum w_i^2$
  - Huber : $Loss_{reg} = Loss + \lambda_h \left(\frac{1}{2} L_2 \;\; if \;\; L1 \leqslant \delta \;\; else \;\; \delta \left(L_1 - \frac{1}{2} \delta \right) \right)$
- Regularization parameters need to be optimized too

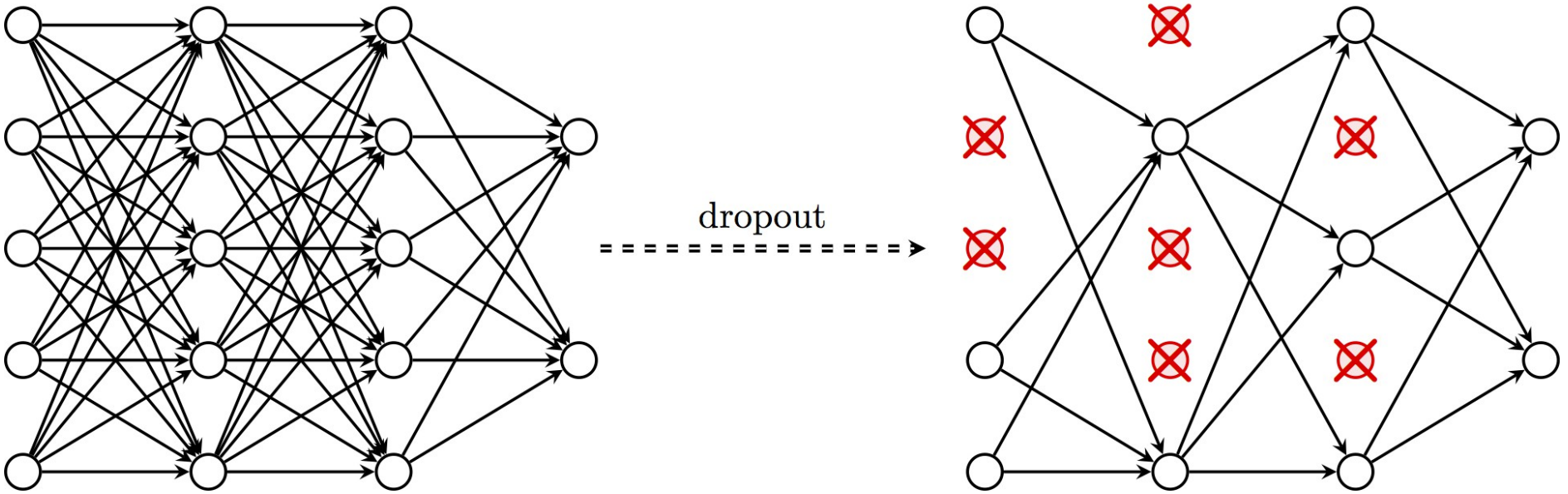
L1 over-constrains to few parameters



L2 explodes with outliers

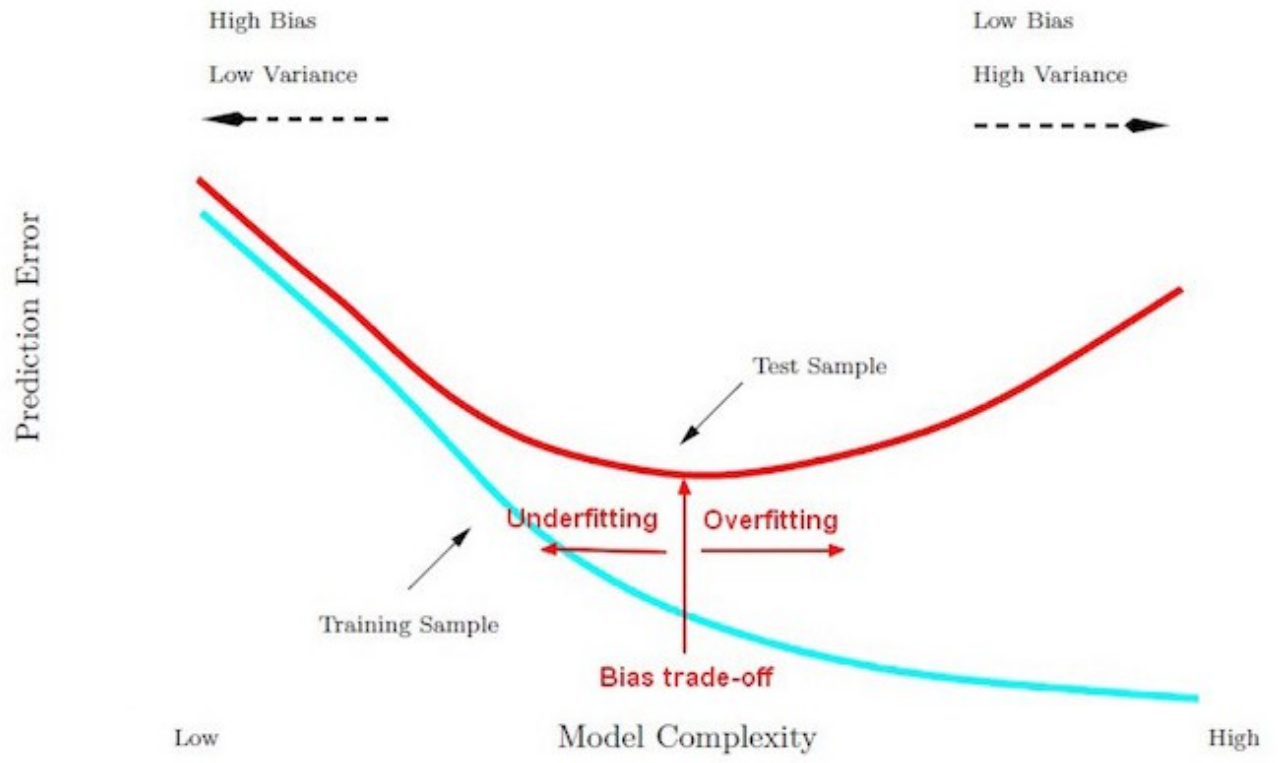
Drop-Out

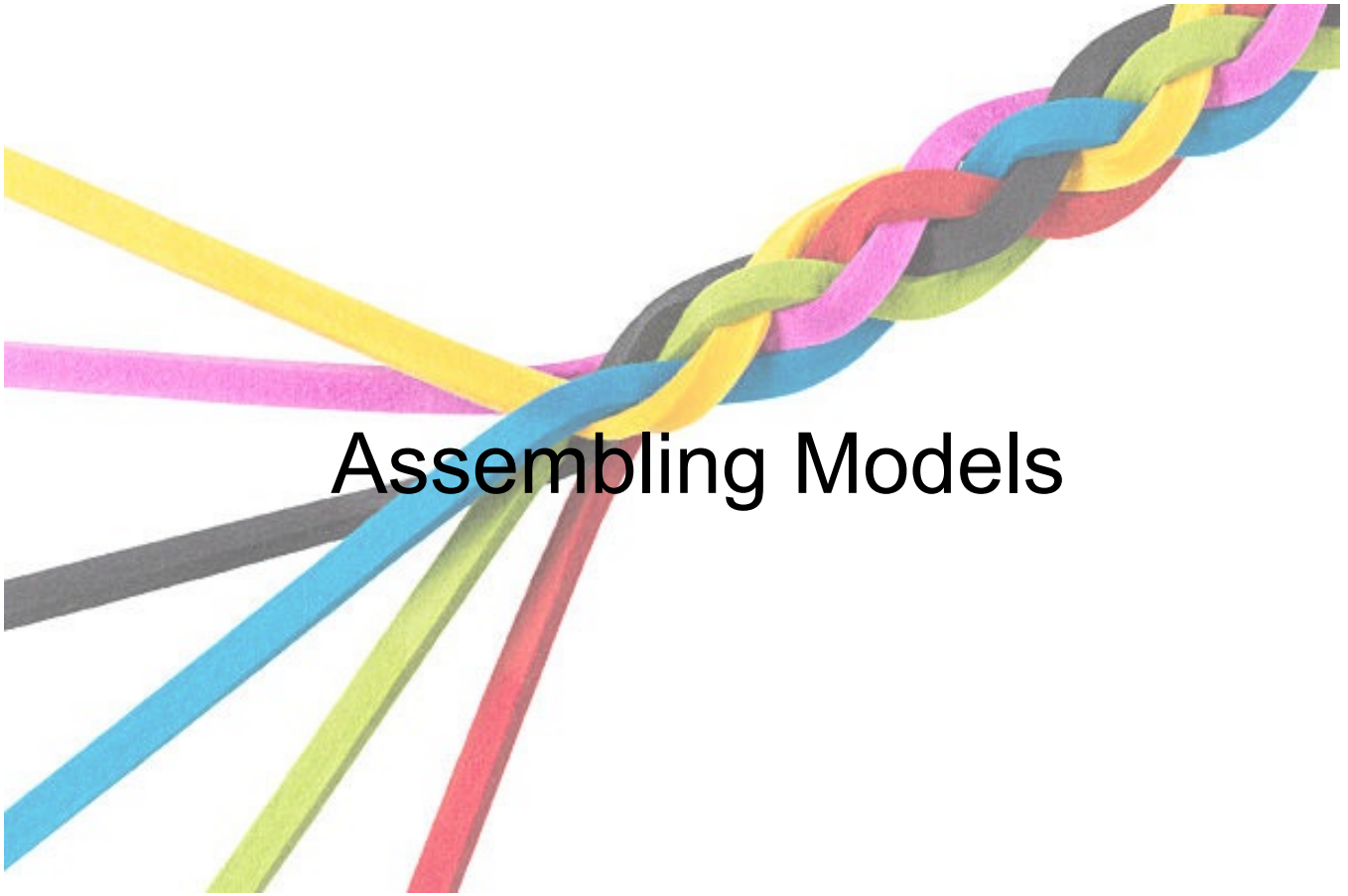
- Special case of regularization in artificial neural network
- During training of the model, set some nodes inactive at a random rate



Generalization

- Systematic error ≡ bias
- Sensitivity of prediction ≡ variance
- A good model is a tradeoff both





Assembling Models

Ensembling

- A single model might have limited performance for various reasons related to how the model is constructed
- Considering output from multiple models can outperform single models
- Loss of interpretability of the final assembled model trade-off of improved performance
- Models can be trained in parallel, sequentially, or both …
  - Boosting
  - Gradient Boosting
  - Bagging
  - Random forest

Logistic Regression

- Logistic regression can be as a simple assembling method
- Linear combination of all models to form a discriminant
- Discriminant used with logistic function to define a per sample probability
- Maximization of cross entropy through regular gradient descent provides the final discriminant

$$H(x) \equiv \sum_j \omega_j h_j(x)$$

$$p(y_i = 1 | x_i) = p_i \equiv \frac{1}{1+e^{-H}}$$

$$p(y_i | x_i) = p_i^{y_i} (1 - p_i)^{1-y_i}$$

$$\omega_{model} = arg\ \underset{\omega}{min} \left(-\log\left(p(y_i | x_i) \right) \right)$$

Boosting

- Adaboost is a method of model assembling such that each model is learning to improve the performance over mis-classified samples
- Train a classifier on the weighted samples
- Define the update to the sequence of classifier

$$M_m(x) = M_{m-1}(x) + \alpha_m h_m(x)$$

$$where \quad \alpha_m \equiv \frac{1}{2} \ln\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$$

$$where \quad \epsilon_m = \sum_{h_m(x_i) \neq y_i} \omega_{i,m}$$

- The weights of the samples are then updated for next step

$$\omega_{i,m+1} = \frac{\omega_{i,m}}{Z_m} e^{-\alpha_m y_i h_m(x_i)} \quad with \quad \omega_{i,1} = \frac{1}{N}$$

- Variations for regression and classifications

Gradient Boosting

- Gradient boosting is a method of model assembling such that each model is learning to improve the performance of the previous one for a provided loss function

$$Loss \equiv L(target\,,\, prediction) = L(t\,,\, p)$$

- The pseudo-residuals *r* captures how much to update the previous output to minimize the desired loss
- Train the next model on the pseudo-residuals $r_{m,i}$
- Find a multiplier for the new model to conclude iteration

$$r_{m,i} = -\left[\frac{\partial L}{\partial p}\right]_{M_{m-1}(x_i)} \qquad M_m(x) = M_{m-1}(x) + \gamma_m h_m(x)$$

$$where \quad \gamma_m \equiv arg\min_{\gamma} \sum_i L(y_i, M_{m-1}(x_i) + \gamma\, h_m(x_i))$$

- Possible variation of the methods for tree-based models
- Applicable to other model types

Bagging

- Bootstrap aggregating, a.k.a bagging assembles multiple models trained from B bootstrap ensembles of the training data

$$f(x)=\frac{1}{B}\sum_{b} f_b(x)$$

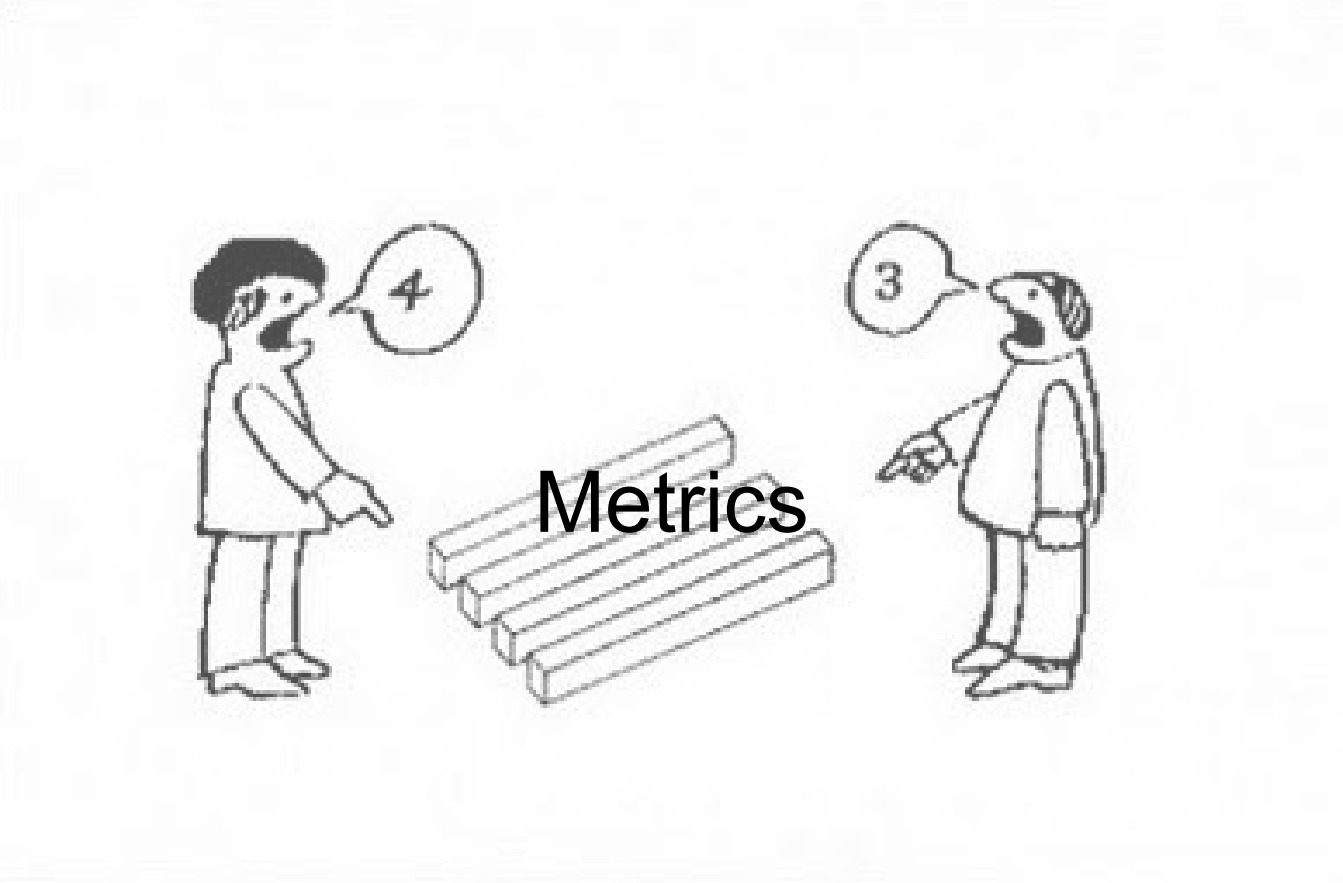
- Reduces the variance of the final model
- Provides an estimate of the uncertainty on the model response

$$\sigma_f(x)=\sqrt{\frac{\sum_{b}\left(f_b(x)-f(x)\right)^2}{B-1}}$$

- Commonly used on decision trees, however applicable to all types of model

Random Forest

- Another ensembling technique to improve performance of decision trees
- Variance from bootstrapping is overestimated because the sub-samples are not purely independent
- Combined bootstrapping and random feature selection
- Feature selection increases the variance of a single tree
- The random feature selection make the decision trees more independent and decrease the variance
- Trade-off in variance often makes random forest a better model

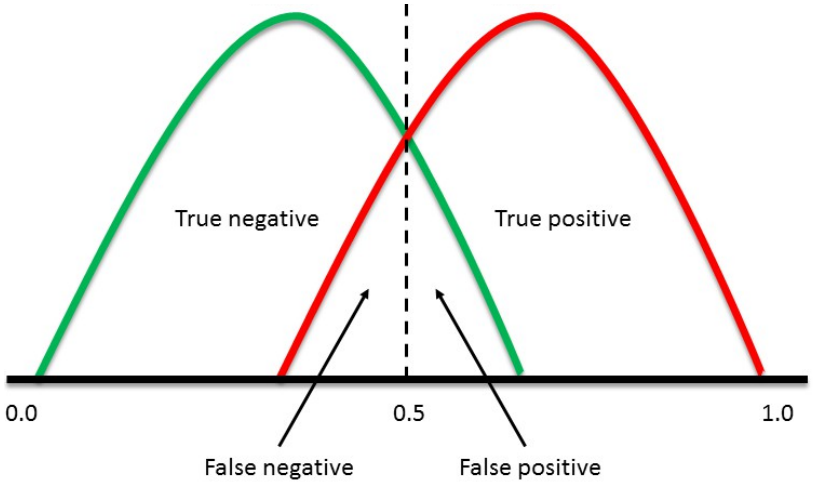

4
3
Metrics

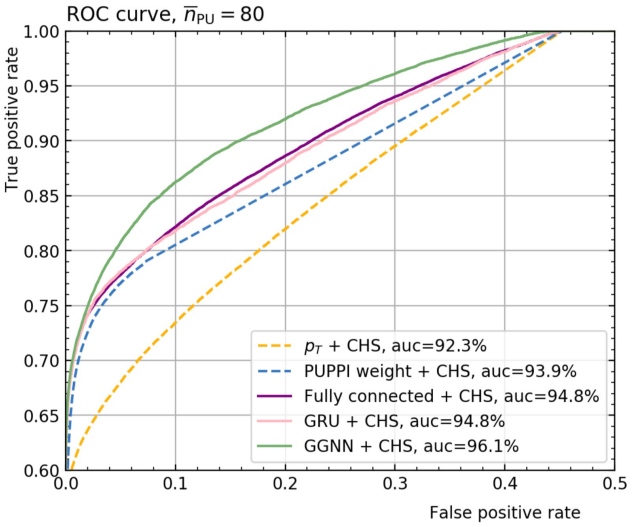
Figure(s) of Merit(s)

- Objective function in optimization might be chosen for computational reason (differentiable, …)
- Objective function might only be a proxy to the actual figure of merit of the problem at hand
- Multi-objective optimization is subject to trade-off between objectives
- While model optimization is based on the loss function over the training set, following the evolution of a more interesting metric over the validation can help selecting models that are better for the use case

Receiver Operating Characteristic

- Characterize the various operating point possible for a given model, depending on the purpose of the model
- Practical visually to compare models
- Below the first diagonal is worse then random guess
- Area under the curve (AUC) a.k.a the integral, of the ROC curve often use as single metric
- AUC is representative when model curve are not crossing





Classification metrics

- Cross entropy : $H(p,q)=E_{p}[-\log q]=-\sum_{x} p(x)\log q(x)$
- Accuracy (acc)
- True positive rate (TPR), recall
- False positive rate (FPR)
- Precision
- F1 score
- Area under the ROC curve

		True condition				
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**, Power	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$	F_1 score = $\frac{2}{\frac{1}{\text{Recall}}+\frac{1}{\text{Precision}}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$		

Regression Metrics

- Mean absolute error : $MAE(x,y)=\frac{1}{N}\sum_i |h(x_i)-y_i|$
- Mean squared error : $MSE(x,y)=\frac{1}{N}\sum_i (h(x_i)-y_i)^2$
- Explained variance : $AV(x,y)=1-\left(\frac{\sigma(h(x_i)-y_i)}{\sigma(h(x_i))}\right)^2$

- If both model prediction and target are distributions
  - KL divergence : $D_{KL}(p||q)=E_p\left(\log\left(\frac{p}{q}\right)\right)$
  - Earth-mover distance :

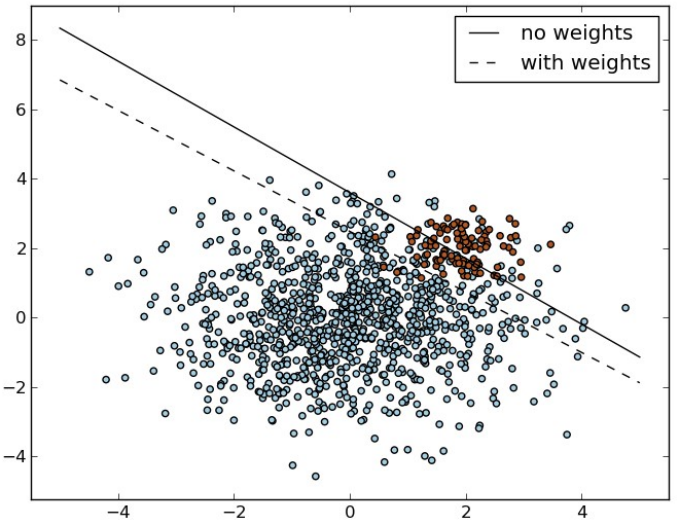
$$EMD(p,q)\equiv W(p,q)=\inf_{\gamma} E_{(x,y)\sim\gamma}[||x-y||]$$

$$s.t.\ \gamma_X=p\,;\gamma_Y=q$$

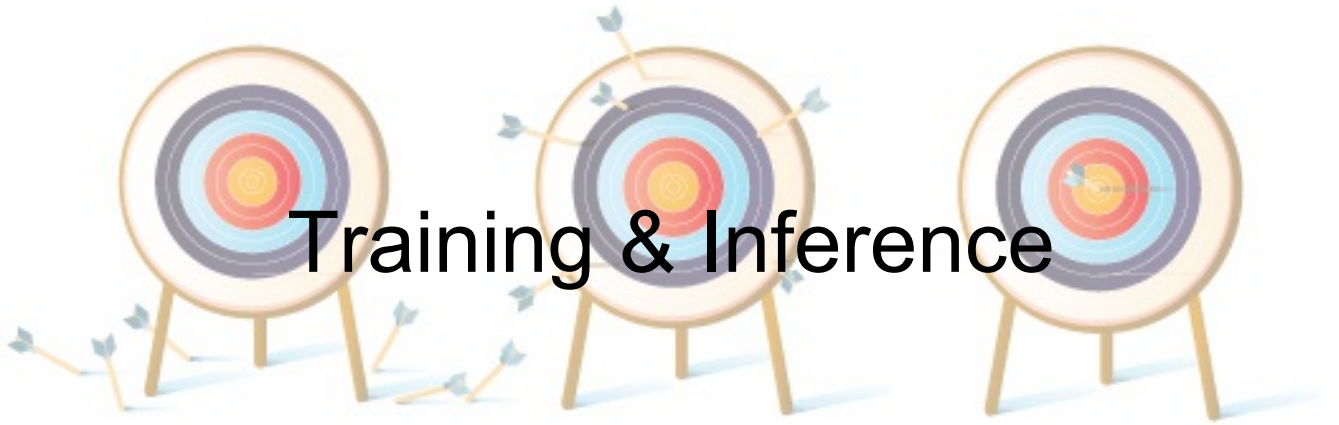
$$\text{in}\quad \gamma(x,y)=\gamma_{X|Y}(x|y)\gamma_Y(y)=\gamma_{Y|X}(y|x)\gamma_X(x),$$

Class Imbalance

- In many cases the number of samples varies significantly from class to class
- Class imbalance biases the performance on the minority class
- Multiple ways to tackle the issue
  - Over-sample the minority class
  - Synthetic minority over-sampling
  - Under-sample the majority class
  - Weighted loss function
  - Active learning



- Metrics can be sensitive to class imbalance and be misguiding if not correct : e.g. 99% accuracy with 0% recall



Training & Inference

Training

- Training phase or learning phase is when the parameters of the model are adujsted to best solve the problem
- For some model/technique (especially deep learning) this can become computationally prohibitive
- General purpose graphical processing units (GP-GPU) offer an enormous amount of parallel compute power, applicable to specific numerical problems
- Matrix calculation, minibatch computation, deep learning, … can get a significant boost from GP-GPU.
- Further parallelization can be obtained across multiple nodes/GPU using

Distributed Training

- Further parallelization can be obtained across multiple nodes/GPU
- Multiple directions of parallelism
  - **Hyper-parameter distribution** : in k-folding or hyper-opt, models are independently trained
  - **Model distribution** : distribute part of the model numerical calculation over hosts
  - **Gradient distribution** : distribute the computation of the gradient over hosts
  - **Batch distribution** : computation over minibatch are independent and later aggregated (staleness issue)

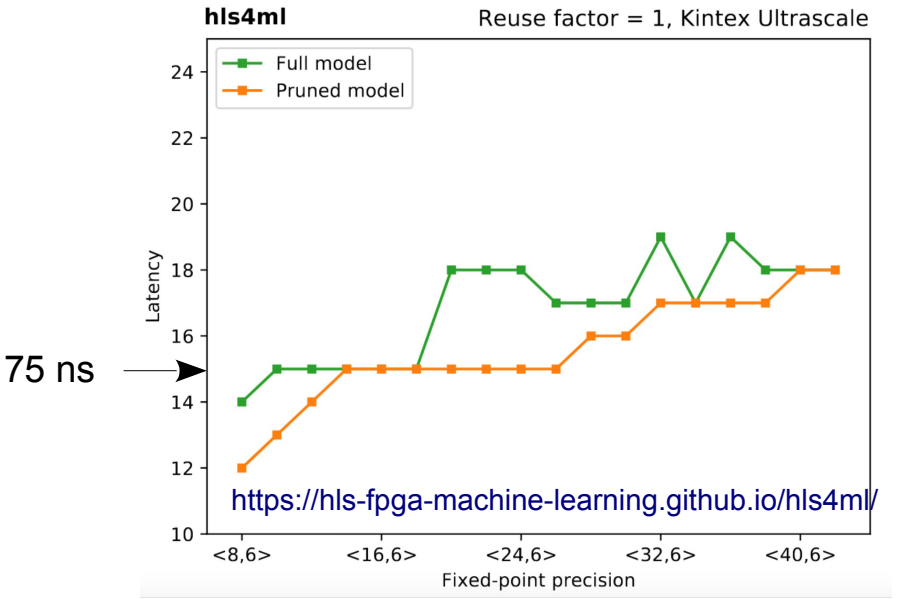
Hyper-parameter Optimization

- Most optimization methods and models require hyperparameters
  - number of layers in an ANN, number of leaves in a decision tree, learning rates, …
- In most cases these parameters cannot be optimized while the model is trained
- Their values can however significantly influence the final performance

- These can be optimize in various ways
  - Simple grid search
  - Bayesian optimization
  - Evolutionary algorithm
- Model comparison should be done very carefully
  - Controlling that performance comparison is fair
  - K-folding is a “must”

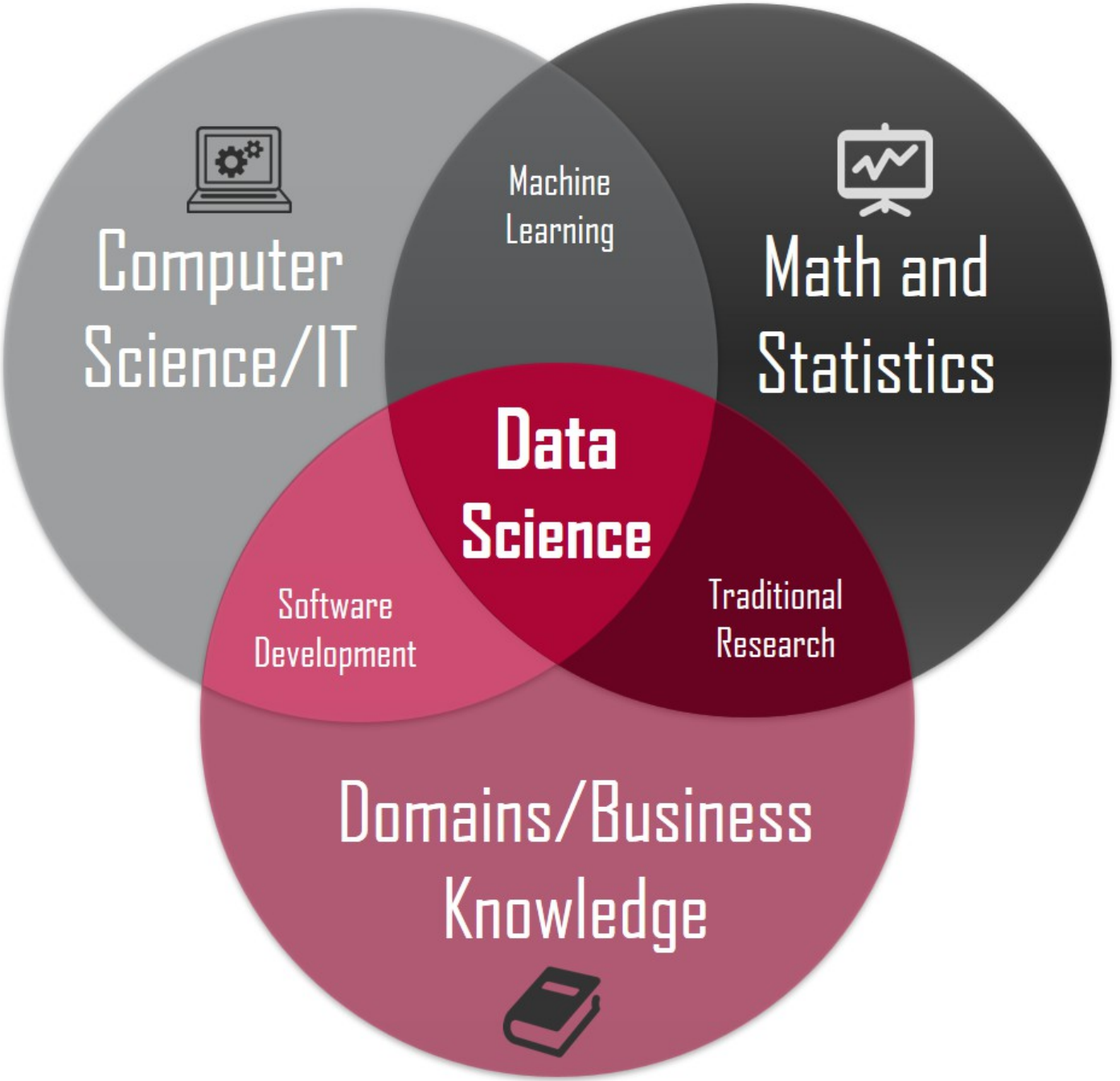
Inference

- Contrary to training, making prediction from a trained model is usually rather fast, even on CPU
- However fast is may be, it might still not be fast enough for the particular application
- Faster inference can be obtained on specialized hardware GP-GPU, TPU, FPGA, neuromorphic, … when the application allows it (trigger, onboard electronics, ...)



Summary/Recap Lecture 1

- Machine Learning is tightly coupled to an optimization problem
- Checking variance in model performance is extremely important
- Large model require lots of data
- Assembling models yields better performance


Computer Science/IT
Machine Learning
Math and Statistics
Data Science
Software Development
Traditional Research
Domains/Business Knowledge