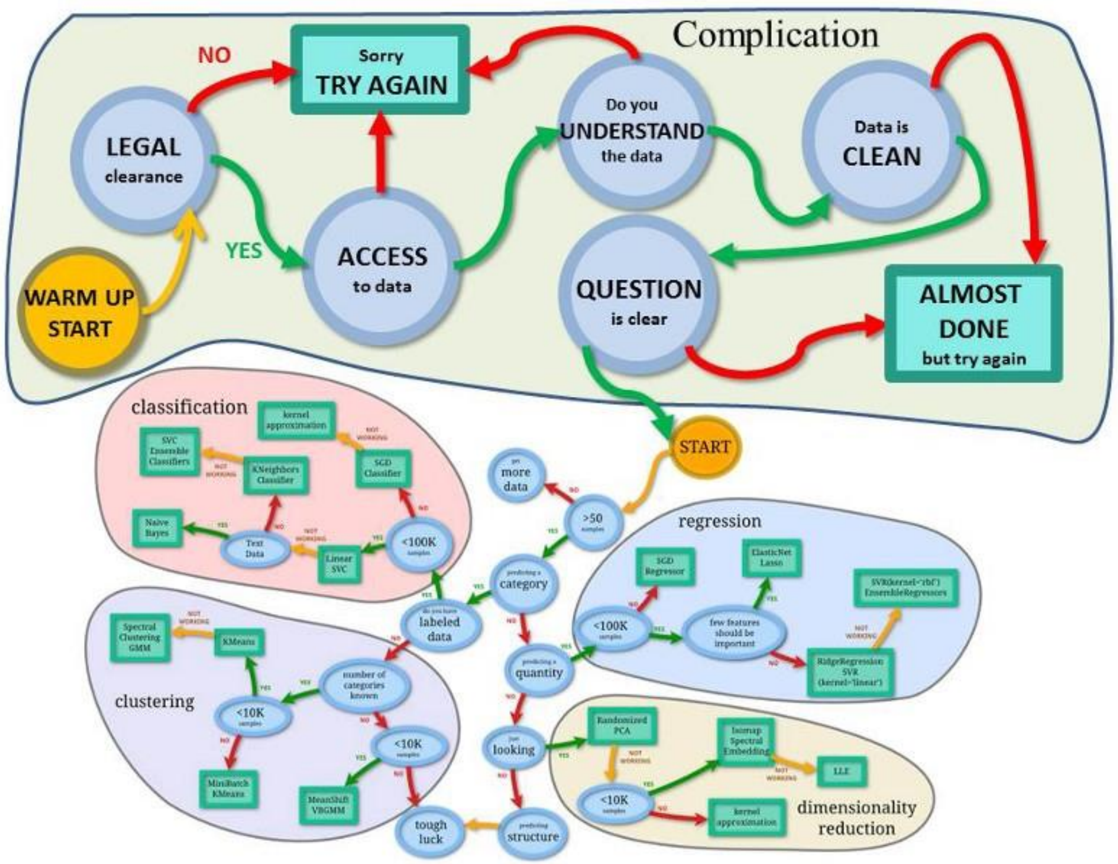
ML Advanced Techniques

Machine Learning II

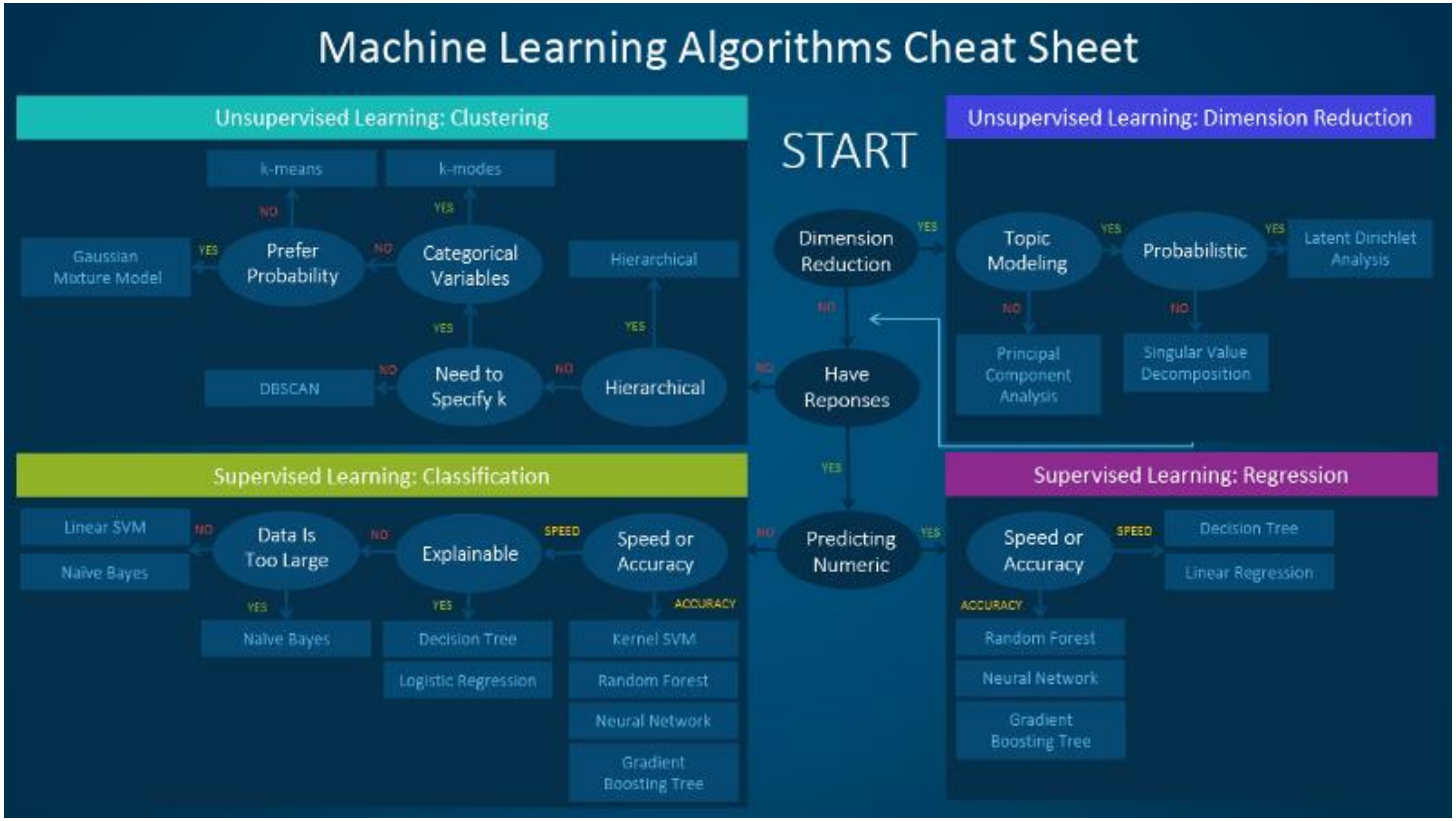
Master in Business Analytics and Big Data

acastellanos@faculty.ie.edu

The Big Picture



Another Big Picture



Choosing an algorithm

Algorithm	Type	Tolerate many features	Parameterization	Memory Size	Data Requirements	Overfiting	Difficulty	Learning Time	Prediction Time	Intepretability
Linear Regression	R	**Weak**	**Simple / Intuitive**	**Small**	**Small / Very Small**	Low	**Weak**	**Weak**	Weak	Good
Logistic Regression	C	**Weak**	Simple	**Small**	**Small / Very Small**	Low	**Weak**	**Weak**	Weak	Good
Decision Tree	R & C	Strong	**Simple/Intuitive**	Large	Small	**Very High**	Weak	Weak	Weak	**Very Good**
Random Forest	R & C	Strong	**Simple/Intuitive**	Very Large	Large	Medium	Average	Costly	**Costly**	Good
Boosting	R & C	Strong	**Simple/Intuitive**	Very Large	Large	Medium	Average	Costly	Weak	Average
Naïve Bayes	C	Weak	NO	Small	Small	Low	Weak	Weak	Weak	Good
SVM	C	Very Strong	Complex	Large	Large	Medium	High	Costly	Weak	Difficult
NN	C	**Very Strong**	**Very Complex**	**Very Large**	**Very Large**	Medium/ High	**Very High**	**Very Costly**	Weak	**Very Difficult**
K-means	C	Strong	Simple	Small	Small	---	High	**Weak**	**Costly**	Average
LDA-QDA	C	Strong	Simple	Small	Small	Low	Weak	Weak	Weak	Good

Some Practical Advices

- Have a look to your data before doing anything!
  - Discover errors
  - Improve your feature engineering
- Work iteratively
  - Begin with simple algorithms whose parametrization is intuitive
- POCs
- There are not magic bullets
  - ML thrives on continual trial and error
- Try them all*
- Cross-validate your evaluation
  - It is test error what matters

Some Resources

- Andrew Ng's Advice for applying Machine Learning
  - https://see.stanford.edu/materials/aimlcs229/ML-advice.pdf
- Use case with regression
  - http://www.dmi.unict.it/farinella/SMM/Lectures/09Dic2015_3.pdf
- Tutorial in Python
  - https://jmetzen.github.io/2015-01-29/ml_advice.html
- An example of Machine Learning Notebook
  - http://nbviewer.jupyter.org/github/rhiever/Data-Analysis-and-Machine-Learning-Projects/blob/master/example-data-science-notebook/Example%20Machine%20Learning%20Notebook.ipynb

Gradient descent algorithm

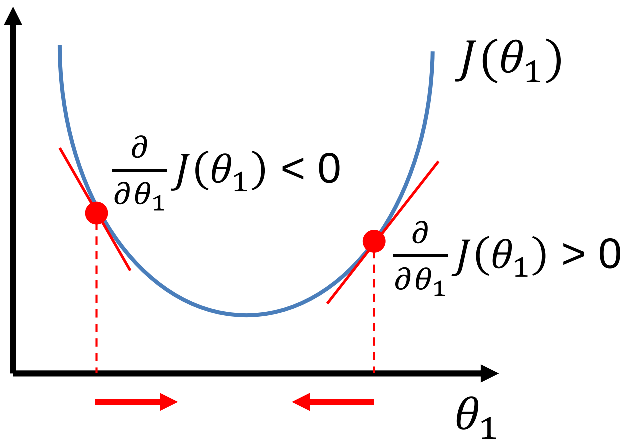
Repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

}, $for\ j = 0, 1$

Derivative

Learning rate



Feature Engineering

Cited by 10.779

Journal of Machine Learning Research 3 (2003) 1157-1182 Submitted 11/02; Published 3/03

An Introduction to Variable and Feature Selection

Isabelle Guyon ISABELLE@CLOPINET.COM
Clopinet
955 Creston Road
Berkeley, CA 94708-1501, USA

André Elisseeff ANDRE@TUEBINGEN.MPG.DE
Empirical Inference for Machine Learning and Perception Department
Max Planck Institute for Biological Cybernetics
Spemannstrasse 38
72076 Tübingen, Germany

Editor: Leslie Pack Kaelbling

Abstract

Variable and feature selection have become the focus of much research in areas of application for which datasets with tens or hundreds of thousands of variables are available. These areas include text processing of internet documents, gene expression array analysis, and combinatorial chemistry. The objective of variable selection is three-fold: improving the prediction performance of the predictors, providing faster and more cost-effective predictors, and providing a better understanding of the underlying process that generated the data. The contributions of this special issue cover a wide range of aspects of such problems: providing a better definition of the objective function, feature construction, feature ranking, multivariate feature selection, efficient search methods, and feature validity assessment methods.

Keywords: Variable selection, feature selection, space dimensionality reduction, pattern discovery, filters, wrappers, clustering, information theory, support vector machines, model selection, statistical testing, bioinformatics, computational biology, gene expression, microarray, genomics, proteomics, QSAR, text classification, information retrieval.

1 Introduction

As of 1997, when a special issue on relevance including several papers on variable and feature selection was published (Blum and Langley, 1997, Kohavi and John, 1997), few domains explored used more than 40 features. The situation has changed considerably in the past few years and, in this special issue, most papers explore domains with hundreds to tens of thousands of variables or features:[1] New techniques are proposed to address these challenging tasks involving many irrelevant and redundant variables and often comparably few training examples.

Two examples are typical of the new application domains and serve us as illustration throughout this introduction. One is gene selection from microarray data and the other is text categorization. In the gene selection problem, the variables are gene expression coefficients corresponding to the

1. We call "variable" the "raw" input variables and "features" variables constructed for the input variables. We use without distinction the terms "variable" and "feature" when there is no impact on the selection algorithms, e.g., when features resulting from a pre-processing of input variables are explicitly computed. The distinction is necessary in the case of kernel methods for which features are not explicitly computed (see section 5.3).

Steps

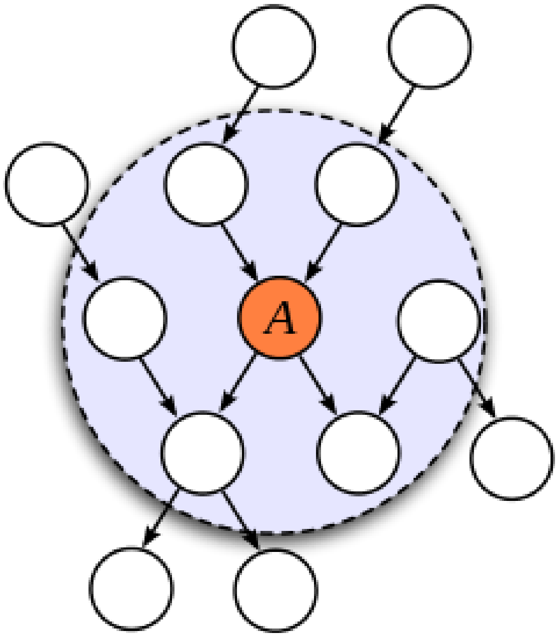
1. Do you have domain knowledge?	Ad hoc features
2. Are your feature commensurate?	Normalize
3. Interdependence of features?	Conjunctive feature
4. Prune input variables?	Disjunctive features
5. Assess features individually?	Variable ranking
6. Suspect data is dirty?	Outliers detection
7. Don’t know what to try first?	Linear predictor
8. Have more ideas and time?	Feature construction
9. Want a stable solution?	Subsample & redo

Conjunctive features

Conjunctive features = features that explicitly represent **combinations of features**, and the other atomic features as primitive features.

Filters are faster than wrappers or embedded.

- Use a **linear predictor** as a filter to later train a more complex predictor on selected features
- Use **Markov blanket** to filter based on information theory (Bayesian networks)



Markov blanket

Steps

1. Do you have domain knowledge?	Ad hoc features
2. Are your feature commensurate?	Normalize
3. Interdependence of features?	Conjunctive feature
4. Prune input variables?	Disjunctive features
5. Assess features individually?	Variable ranking
6. Suspect data is dirty?	Outliers detection
7. Don’t know what to try first?	Linear predictor
8. Have more ideas and time?	Feature construction
9. Want a stable solution?	Subsample & redo

Disjunctive features

Optimally Extracting Discriminative Disjunctive Features for Dimensionality Reduction

Amrita Saha
IBM Research
Bangalore, India
amrsaha4@in.ibm.com

Naveen Nair
Indian Institute of Technology,
Bombay, India
naveennair@cse.iitb.ac.in

Ganesh Ramakrishnan
Indian Institute of Technology,
Bombay, India
ganramkr@cse.iitb.ac.in

Use features derived from the original input

- **Clustering**
  - Replace or add groups of similar variables by a **cluster centroid**
  - Information Bottleneck: $J = I\left(X, \tilde{X}\right) - \beta I(\tilde{X}, Y)$
- **Linear transformations:** PCA, LDA, SVD, PLS
- **Spectral transformations:** Fourier, wavelets, convolutions, kernels
- **Simple functions:** products or monomials

Steps

1. Do you have domain knowledge?	Ad hoc features
2. Are your feature commensurate?	Normalize
3. Interdependence of features?	Conjunctive feature
4. Prune input variables?	Disjunctive features
5. Assess features individually?	Variable ranking
6. Suspect data is dirty?	Outliers detection
7. Don’t know what to try first?	Linear predictor
8. Have more ideas and time?	Feature construction
9. Want a stable solution?	Subsample & redo

Variable ranking

The Feature Selection Problem : Traditional Methods and a New Algorithm

Kenji Kira
Computer & Information Systems Laboratory
Mitsubishi Electric Corporation
5-1-1 Ofuna, Kamakura
Kanagawa 247, Japan
kira@sy.isl.melco.co.jp

Larry A. Rendell
Beckman Institute and Department of Computer Science
University of Illinois at Urbana-Champaign
405 N. Mathews Avenue
Urbana, IL 61820, U.S.A.
rendell@cs.uiuc.edu

You've already done this with:

- **Score** each feature according to its χ^2 or correlation coef.
- Or **predictive power measured in terms of the error rate**
  - Vary **thresholds** to find the balance between *fpr* and *fnr*.
- **Relief** algorithm
  - The closest example of the same class (nearest hit) and the closest example of a different class (nearest miss) are selected. The score $S(i)$ of the i_{th} variable is computed as the average over all examples of magnitude of the difference between the distance to the nearest hit and the distance to the nearest miss, in projection on the i_{th} variable.

Steps

1. Do you have domain knowledge?	Ad hoc features
2. Are your feature commensurate?	Normalize
3. Interdependence of features?	Conjunctive feature
4. Prune input variables?	Disjunctive features
5. Assess features individually?	Variable ranking
6. Suspect data is dirty?	Outliers detection
7. Don’t know what to try first?	Linear predictor
8. Have more ideas and time?	Feature construction
9. Want a stable solution?	Subsample & redo

Feature Construction

Feature Construction Methods: A Survey

Parikshit Sondhi
Univeristy of Illinois at Urbana Champaign
Department of Computer Science
201 North Goodwin Avenue Urbana, IL 61801-2302
sondhi1@uiuc.edu

We want:

1. Generate a set of features that help improve prediction accuracy.
2. Are computationally efficient.
3. Are generalizable to different classifiers.
4. Allow for easy addition of domain knowledge.

Logic Programming (FRINGE)

- In each iteration, new features are constructed by combining pairs of features in the existing feature space using *negation* and *and* operators.

Genetic Programming

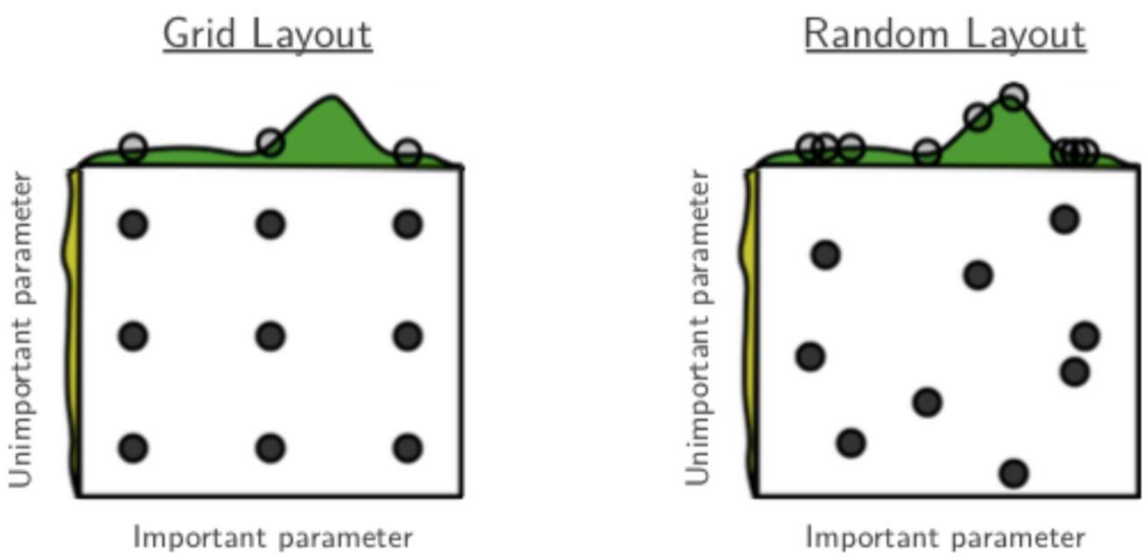
Bootstraps

Goal is stabilize variance

- The feature selection process is repeated with sub-samples of the training data.
- The union of the subsets of variables selected in the various bootstraps is taken as the final “stable” subset.
- This joint subset may be at least as predictive as the best bootstrap subset.
- Analyzing the behavior of the variables across the various bootstraps (how frequently they appear in the bootstraps) also provides further insight

Bayesian Optimization

- Grid search
- Random search (better)



Bayesian Optimization

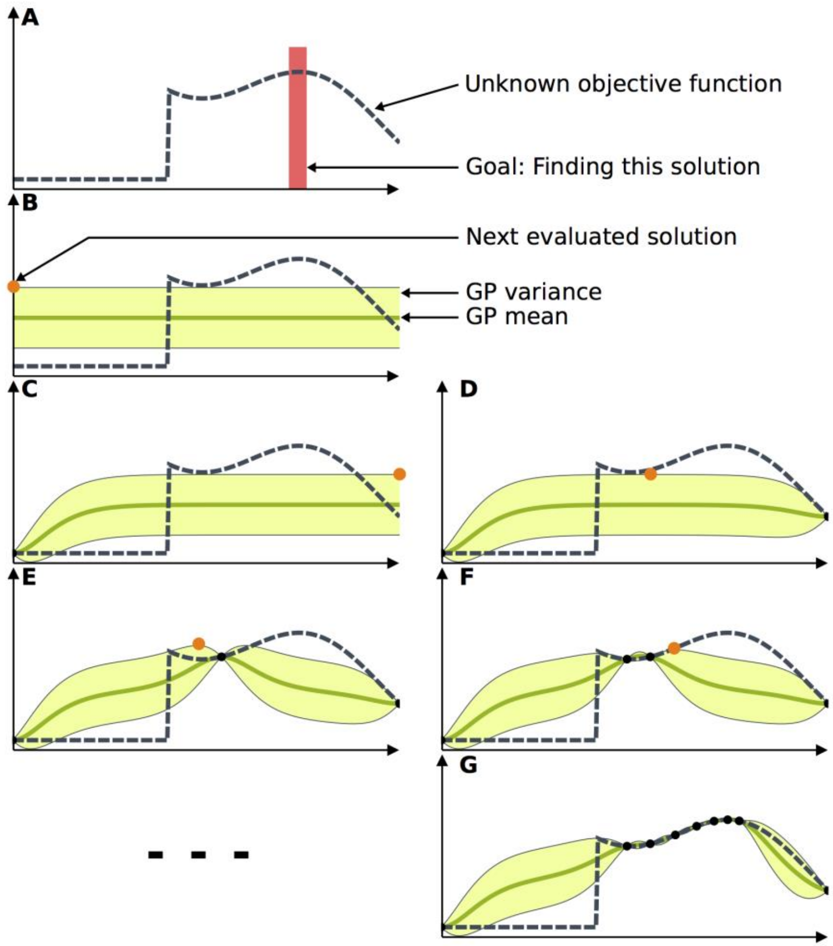
(A)*The goal of this toy problem is to find the maximum of the unknown objective function.*

(B)*The Gaussian process is initialized, as it is customary, with a constant mean and a constant variance.*

(C)*The next potential solution is selected and evaluated. The model is then updated according to the acquired data.*

(D)*Based on the new model, another potential solution is selected and evaluated.*

(E)(G) *This process repeats until the maximum is reached.*

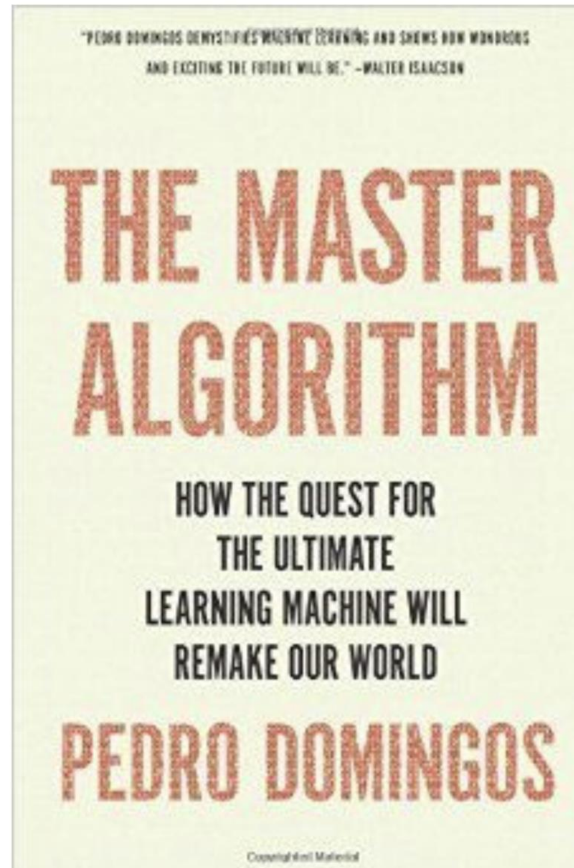


http://www.resibots.eu/limbo/guides/bo.html

Ensembles

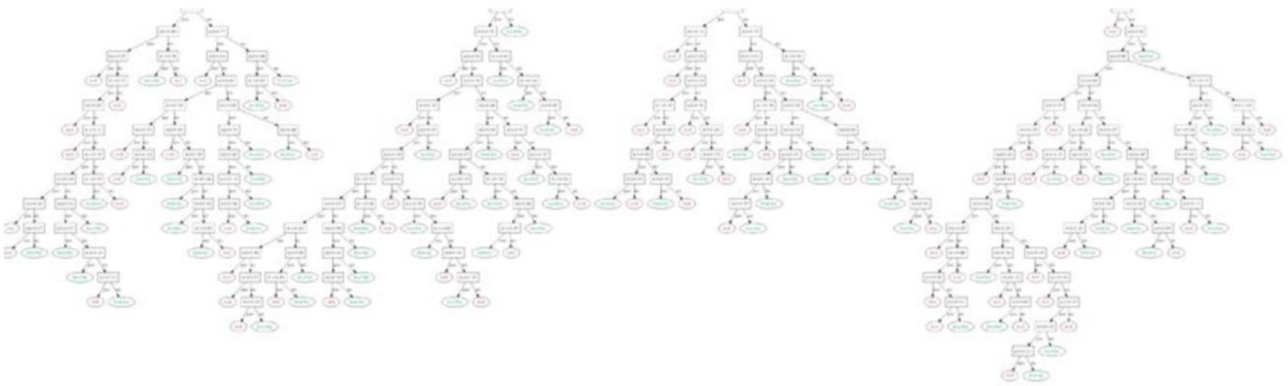
The Master Algorithm?

It definitely is an <u>ensemble!</u>



Ensembles & Feature engineering

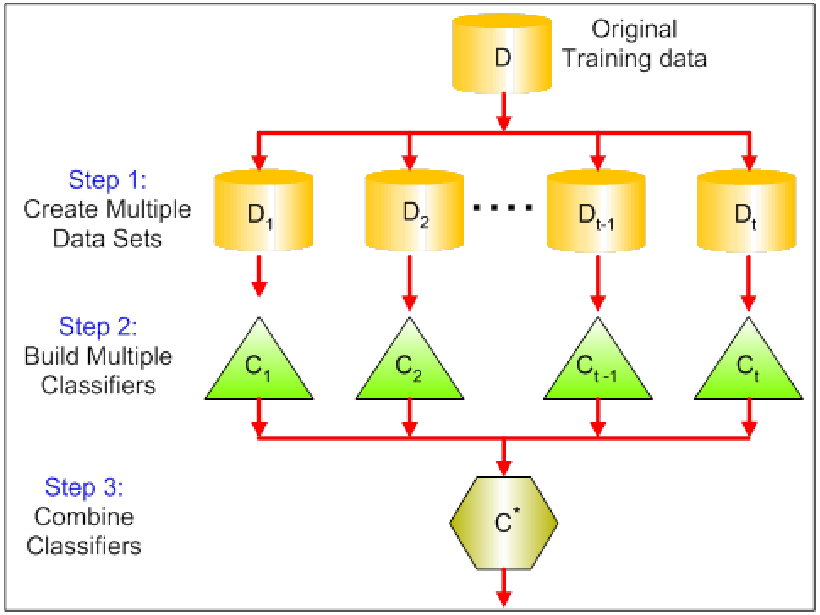
- Ensembles are the way to **turn any model into a feature**!
- E.g. Don't know if the way to go is to use Factorization Machines, Tensor Factorization, or RNNs?
  - Treat each model as a "feature"
  - Feed them into an ensemble



Ensembles: bagging

Bagging (**b**ootstrap **agg**regat**ing**): reduce variance

- Bagging uses **bootstrap** sampling to obtain the data subsets for training the base learners.
- For aggregating the outputs of base learners, bagging uses **voting** for classification and **averaging** for regression.

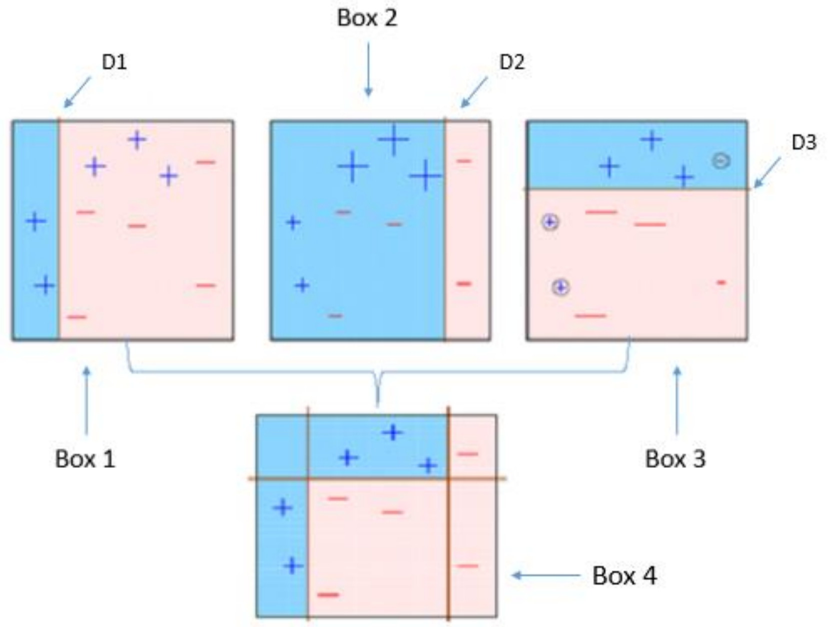


https://www.analyticsvidhya.com/blog/2015/08/introduction-ensemble-learning/

Ensembles: boosting

Boosting aims to turn a set of weak learners into a strong learner algorithm (reduce bias).

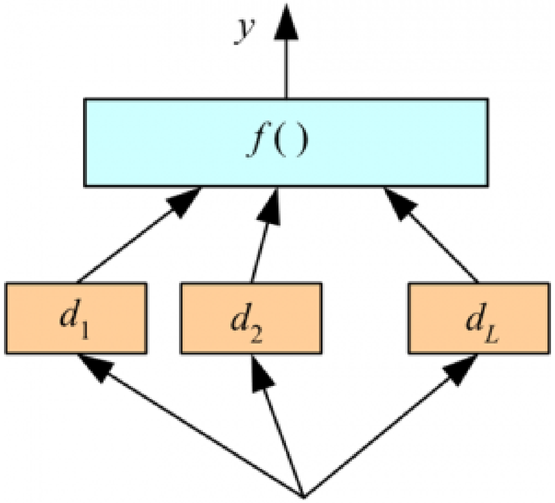
- AdaBoost (Adaptive Boosting)
- Gradient Tree Boosting
- XGBoost



Ensembles: stacking

Stacking = meta ensembling

- A model *stacked* on top of the other models, is used to combine output from different learners.
- This can lead to decrease in either bias or variance error depending on the combining learner we use.



http://blog.kaggle.com/2016/12/27/a-kagglers-guide-to-model-stacking-in-practice/