Topic:

"Story about decision trees based on lectures from one professor"

Konstantin Burlachenko

-kburlachenko@nvidia.com

-bruziuz@stanford.edu

-burlachenkok@gmail.com

References

[1] Classification And Regression Trees, 1983 (Brieman, Friedman, Olshey, Stone) – CART

[2] The Elements of Statistical Learning Book. (Friedman, Tibshirani, Hastie)

[3] More detailed notes then slides https://sites.google.com/site/burlachenkok/articles/decision-trees-parti-decision-trees-for-regression

Why Decision Trees?

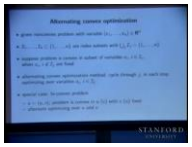
- Neural Nets for some reasons are best (empirically) on supervised problems involved unstructured data
- Tree ensembles (empirically) are very good for structured data on supervised problems.
- Some areas of Human activity need explanation. There are not a lot of models in AI/ML which allow todo it. Decision trees enough powerfull model which allow **todo it**.
- Decision Tree can do something what other technics (like NN) can not:
  - Handle missing input (in natural way)
  - Handle categorical variables (in natural way)
  - Use model in simple way without any big team of DataScience/ML scientist in your company (if you do not have money to hire him/her)
  - Trees has **one meta-parameter** and neural nets has **hundreds of meta-parameters**

Several quotes before begin

"Here is a technic and it works better on this example and it works better always. You should be careful. It's not true." – J.H. Friedman

"For non-convex optimization beside local minimums there is another statistical problem with unstable solution due random nature of samples" - J.H.Friedman

"In non-convex optimization there is a big room for personal expression. Maybe your method works better for some problem, maybe not." – S.P.Boyd
(https://youtu.be/cHVpwyYU_LY?t=646)



Plan for this presentation

This material is based on STATS315B course with J.H.Friedman from Stanford University.

The plan is:

1. About supervised Machine Learning
2. Search strategy
3. Bias-Variance tradeoff
4. Building blocks for Decision Trees
5. Decision Tree for regression
6. Decision Tree for regression 3 main questions and final CART algorithm
7. Summary: Advantages of Trees and Disdvantages of Trees
8. Appendices. Missing data and variable importance

1.What supervised ML is "doing"

Goal is reconstruct $X \rightarrow Y$ in some **approximate** way (e.g. price for stocks for tomorrow)

1. What is really happens we build approximation $\hat{F}$ based on data
2. We assume that we don't know nothing about underlying phenomen (it's simulatanously good and bad)
3. *"Nobody extract functional form,all the we do in this business called AI/ML is approximations " – J.Friedman*
4. *"You can use DL for Vision system and other very difficult problems, but not for problems that we know how to solve exactly. It has no sense to me. I hopes it will be change from several years and people start understand it very soon"* - 2018, S.Boyd *(https://youtu.be/U41e7hKAAPQ?t=6226, 01:43:45)*

1. What is Supervised Machine Learning

1. Model (or pattern) structure

$\hat{F}(x) = \hat{F}(x;a) \in \mathcal{F}(a)$ *(Parametric way to define class of functions. Alternative non-parametric way)*

2. Score criteria

$L(y,\hat{y})$ *is a scalar-value functions of two scalar arguments called* ***loss****. Usually:*

- *It states how we unhappy when real value* ***y*** *and we predict* $\hat{\boldsymbol{y}}$
- *L has minimum when* $\hat{y}$*=y. Also p*eople usually use some well-known loss
- *Via function L we create* ***prediction risk on population/in data***

name	*Math form*
prediction risk	functional in form $\boldsymbol{R}(F) = E_{xy}L(y,F(x))$)
optimal or target function $\boldsymbol{F^*}$	$F^* = argmin_F R(F)$
score on population	$\boldsymbol{S(a)} = E_{xy}L(y,F(x;\boldsymbol{a}))$
score on data. Often $\boldsymbol{l \neq L}$	$\widetilde{S(\boldsymbol{a})} = \frac{1}{N}\sum_{i=1}^{N} l(y,F(x;\boldsymbol{a}) + \lambda P(a)$ (P(a) allow incorporate knowledge about model parameters)

3. Search strategy. The most important thing.

$a = argmin_a \widetilde{S(\boldsymbol{a})}$ *And arised optimization problem can be non-convex.*

"it's more easy to define things then to solve things" – J.Friedman

1. Some statistical problems around ML

1. $X \to Y$ is not a real dependency, but in fact $X, \boldsymbol{Z} \to Y$

2. When we fixed X there is distribution of $Y|\mathrm{X}$. But **nobody predicts it** (what is predicted is only E[Y|X])

3. Is distribution $Y|\mathrm{X}$ it symmetric – **nobody knows**. But everybody assumed this.

4.Is $\mathrm{D}[Y|\mathrm{X}]$ depends on X – maybe. But it's very hard to model – **nobody go into such hard thing**.

5. J.Friedman mentioned that both (2,3,4) are not even remotely true if look too closely

6. We **hope future looks like the past** (and nothing have been changes inside model) and we can use cross-validation to estimate parameters.

- Z behavs is uncontrolling – and it's just "fluctuation" for sample
- Something inside (X,Y) distribution have changed - *concept drift*

1. Some engineering/math problems around ML

- Usually we fit not to real samples of signal, but really into (signal + noise)
- We in general shouldn't believe a lot to provided data from user
- Complexity(size) of function space from which we will pick $X \rightarrow Y$ can affect into quality of model. If we can perfectly "fit provied data" it's reasonable to assume that function space is BIG. And it lead to some troubles as we will see

1. Some math optimization problems around ML

1. If to be honest nobody knows how to deal with non-convex optimization in *some general* way (we know how solve near 5 non-trivial non-convex problems). But Decision Trees or Neural Nets (even with simple transfer/activation functions) is not part of it.

2. If to be honest convex optimization can be solved effectively, but solving *Stochastic Convex Optimization* in open research topic.

3. Combinatorial optimization is even more hard (computationally). Unfortunately there are no general methods when boolean/integer variables come into play which always work in polynomial time (even for simple problems like Integer LP) and solves problem exactly.

4. A.V.Gasnikov mentioned in one of his videos from mathnet that nobody have built any kind of theory on (sub)gradient for non-convex functions

5. Global Optimization technics for non-convex optimization (like *Branch and Bound*) are exist and are based on convex optimization. But they are very slow. In worst case they take computation time $\sim 2^{number\_of\_variables}$

1. What is Machine Learning

1. Methods developed in Machine Learning originally was heuristics
2. *"This method of study phenomens is terribly successfull" - J.Friedman*
3. *"Theory is very thin, but engineers and computer science does not require deep theory too much" – J.Friedman*
4. *"Some methods need years of work to apply them in perfect way like NN" – J.Friedman*
5. *How people understand what method is good? Answer: Totally empirically.*
   - *It lead to fact that we should not over-interpret results*
   - *Situation depends: samples size, signal/noise ratio, target function F^**
6. No free lunch result of David Wolpert and William Macready.

"If try to apply two prediction schemas for all possible target functions and average the result then they will be equivalently bad"

2. Search strategy

Component which allow to find function which minimizes score on available data. From point of view of J.Friedman the real problem is with that *"Search strategy is important. Without it defined previous steps has no sense"* – J.Friedman

1. Unfortunately in case of decision tree and neural nets - arised optimization problem is **non-convex** in parameters(variables)
2. One possible is global optimization but in business of Machine Learning people mostly use greedy methods
3. Especially computation should be feasible in time
4. In some problems like decision tree we in are in situation when objective is highly non-continuous in parameters. And any derivative based method is not applicably in such cases.

2. Search strategy/about non-convexity

If objective is non-convex there are more problem beside local minimums.

1. In convex optimization you can start wherever you want
2. Data is random samples
3. Score is a function of sample too, so it's a random variable.
4. And minimum score is random variables
5. If score function (objective) is convex – we're in good shape. Optimal point will not perturb too much if data perturb not too much.

And it's not a case for non-convex functions. Shifted datasets with similar starting point intuitively is the same as non-shifted datasets with shifted starting point.

Non-convexity:

"In non-convex optimization methods there are a lot of room for personal expression" – S.Boyd

"In non-convex optimization any classification of problems just does not exist right now, even there are a lot of possible situations" – B.Polyak

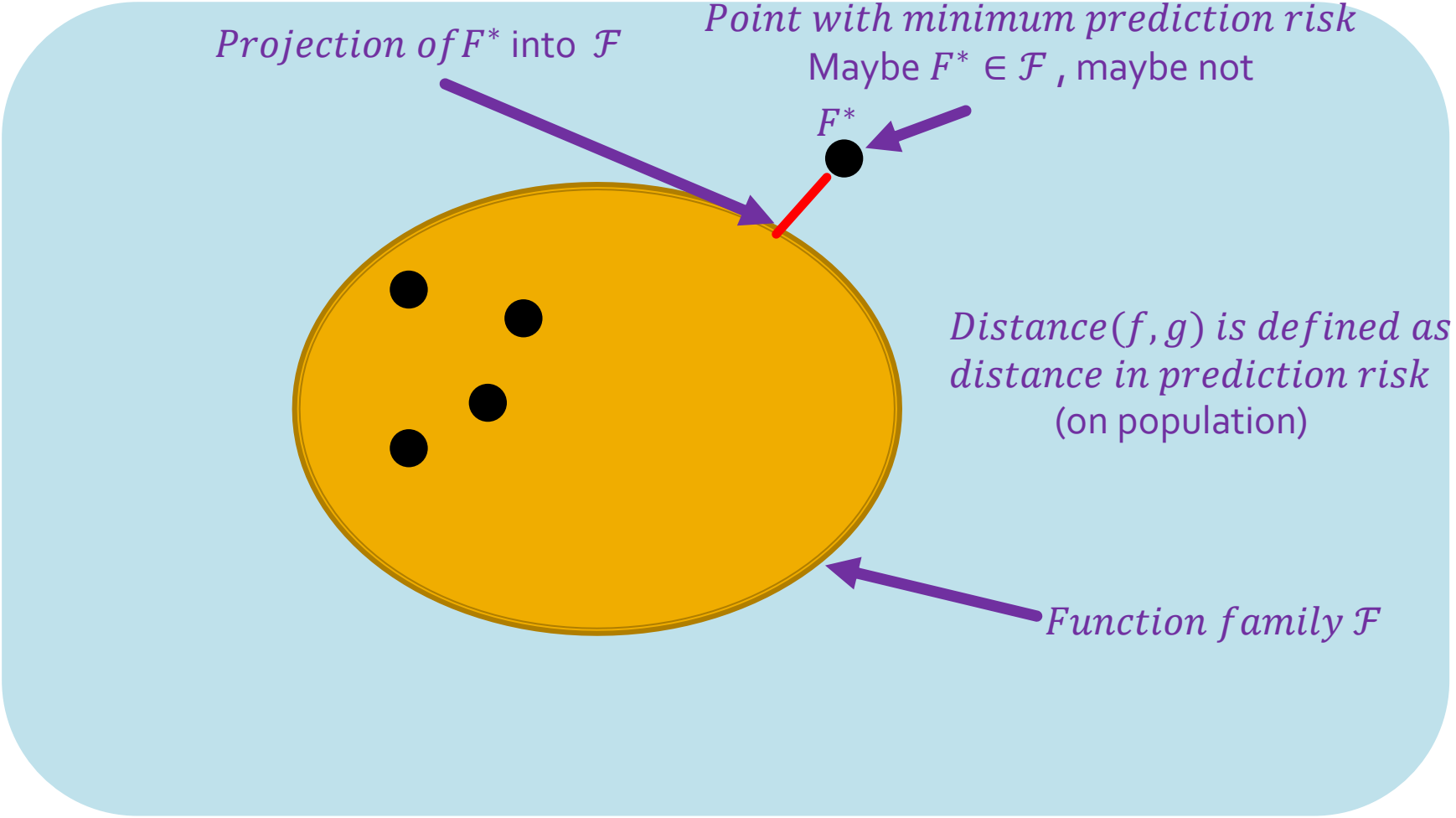
3. Bias-Variance tradeoff

For any infinite set M we have: $MxM \sim M$.
So any point M^k can be considered as point in two dimensional paper in M^2 after apply bijective mapping.

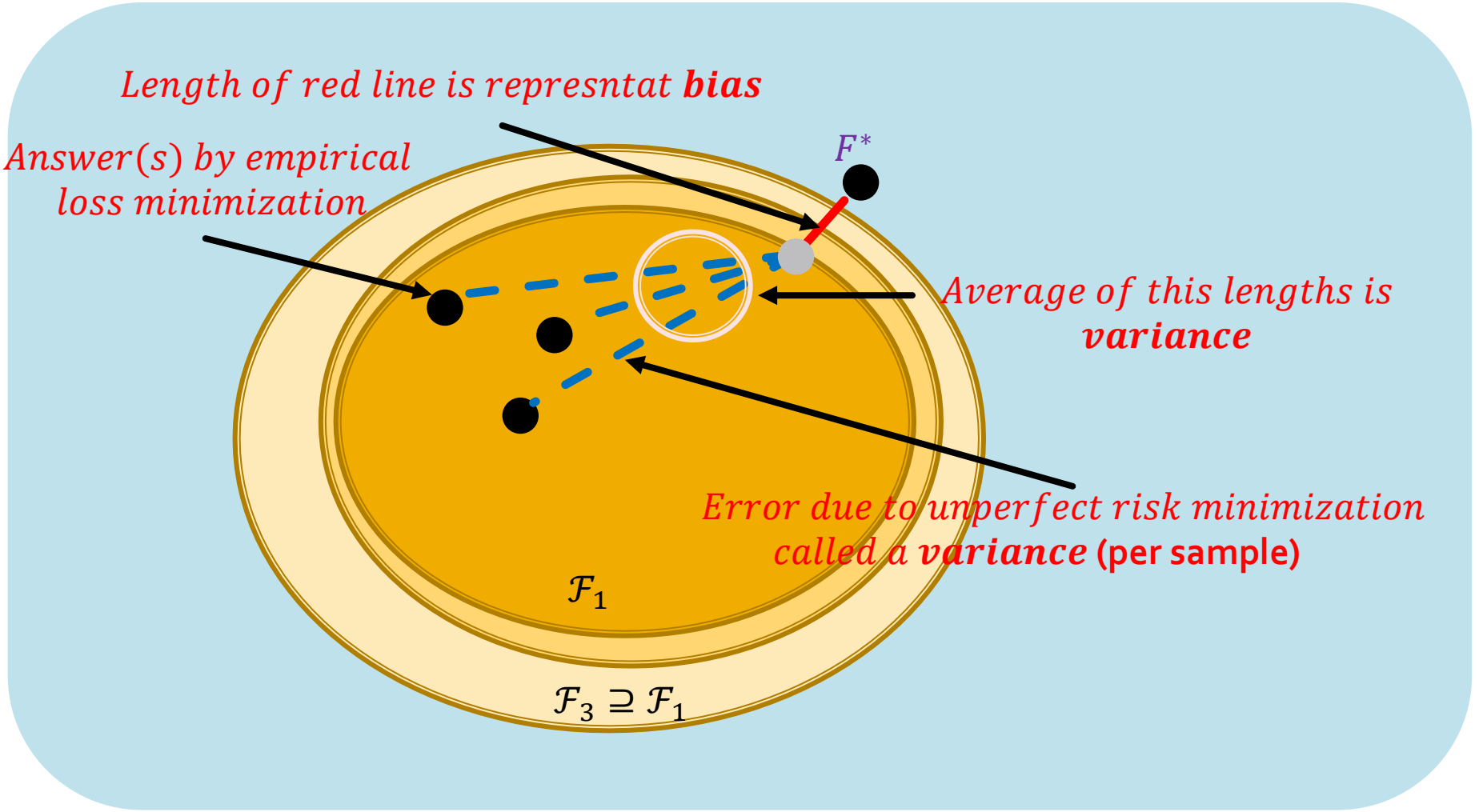
All possible functions $\mathbb{R} \to \mathbb{R}$ can not be considered as point in paper. But lets' be sloppy and assume it can be. So I can "draw" *functions* in $\mathbb{R}^2$…

$\mathcal{F}$

3. Bias-Variance tradeoff



3. Bias-Variance tradeoff



3. Bias-Variance tradeoff

So there are always two kind of errors in this business:
Variance error and **Bias error**:

Reason of variance
we do not know population and we use only data. Variance can also "be corrected" by more amount of data.
It is about uncertanty which function is the best in case of limited amount of data.

Reason of bias
Is that our function class not necessary contain target function. Bias can be fixed by consider more big function space.

In Machine Learning supervised methods:
Complete picture is that we consider series of nested function families and try to pick the best function via cross-validation.

Complete topics and plan for future

1. About supervised Machine Learning
2. Search strategy
3. Bias-Variance tradeoff
4. Building blocks for Decision Trees
    1. History
    2. X decoupling, Data Matrix, types of x_i
    3. Size of train set
5. Decision Tree for regression
    1. Search Strategy(opt.problem solving)
    2. Final search strategy
    3. Why partially constant function is called a tree?
6. Decision Tree for regression 3 main questions and final CART algorithm
    1. Split region as opt.problem
    2. Q1:Which regions select to split within M regions?
    3. Q2: In which variable within a region perform a split ?
    4. Q3: When stop splitting?
    5. CART. Final algorithm
    6. CART. How find cp
7. Summary: Advantages of Trees and Disdvantages of Trees
8. Appendices. Missing data and variable importance

4. Decision Trees/History

- Idea is reinvented every 30 years
- *"Approximation function in high dimensional so hard that even simple decision trees are competitive" - Jerome H.Friedman*

- *The most popular variations this day:*
  - ***CART (1983) from stats community (L.Breiman, J.Friedman, R.Olshen, C.Stone)***
  - CHAID (1979)
    (developed by Morgen, Sorquist, Messenger.*)*
  - *C 4.5 (1993)/C 5.0(1997)*
    Like CART, but come from ML community.
    (All have been done by Ross Quinlan who also developed ID3)

4. Building blocks for Decision Trees/X decoupling

We want to construct function $X \to Y$ which approximated real phenomen.

We should take "care" what is X, but real goal is not "analyze X",
but real what we will do:

(1) Introduce some span of functions
(2) Find a way to obtain coefficients of span expansion

But let's talk about **X**

1. Most time X van be decoupled it as Cartesian product of x_j such that $X = x_1 \times x_2 \ldots \times x_n$

2. Decision tree and Neural Nets naturally works with attribute-value data. It's not only one kind of data.

   Other kind is proximity data and there are only two methods to work with it in Machine Learning – **Nearest Neighboors, Kernel methods**.

4. Building blocks for Decision Trees/Data Matrix

x_{ij} - value of j-th attribute for i-th object.

Attribute #1	Attribute #2	…Attribute #n	Y
x_{11}	x_{12}	x_{1n}	y_1
…	…	…	…
x_{N1}	x_{N2}	x_{Nn}	y_N

This matrix is called "design matrix"|"data matrix"|"flat file" | "spreadsheet"

n is number of columns –
(a.k.a number of "**measurement**"|"**attributes**"|"**variable**" | "**fields**")

N is number of rows –
(a.k.a. "**objects**"|"**samples**"|"**observations**"|"**examples**" in database)

It's only terminology, but still each community use it's own dialect

4. Building blocks for Decision Trees / Size of train set

- Even N is large like 10^6 the more usefull information in *train set* is information about event you try to predict. (Examples with not often events: *click on add*, *crash of airplane*, *crash of nuclear plant*)

- Statistical point of view into "size" of the set is the number of observation which are relative to what you try to predict and not to
  all available observation

- This days data is grabbed by computer and datasets and predictor variables is big like $10^1 - 10^6$

4. If $X = x_1 \times x_2 \ldots \times x_n$ what is x_i ?

Name	Has order x < y?	Has distance metric d=(x,y)=|x-y| ?	Examples
interval scale	+	+	Any real variable
circular/periodic	-	+ minimum length of two possible arcs on circle	Direction, time of days…
ordinal	+	- We can introduce scoring- but it's artificial step	Grades, Size, Number of starts.
categorical/ factorial	-	- or very lightly "+" because d(x,y)= 0 if $x = y$ 1 if $x \neq y$	Word, Language.. It's just a name

- Different procedures accept different type of variables.
- J.Fr. Asked asked carefull when people state "can handle"
- Tress can handle circular variables, even J.H. do not know any implementation which do it

5. Decision Trees, regression case

Every Machine Learning technic has 3 components as we discussed:

1. *Model (or pattern) structure*
2. *Score criteria*
3. *Search strategy*

5. Decision Trees, regression/Model

Model (or pattern) structure. Most general

The structure model for decision trees is the following:

$$F_c(x) = \sum_{m=1}^{M} c_m I(x \epsilon R_m)$$

Where:

I is indicator function in the following form:

$$I(expr) = \begin{cases} 1 \; if \; \text{expr} \; is \; \boldsymbol{true} \\ 0 \; of \; expr \; is \; \boldsymbol{false} \end{cases}$$

Parameters in this models (**it's called variables in optimization community**):

1. $\{R_i\}$ - subregions of input space.$i = \overline{1, m}$
2. c_i - function value on this subregions. $i = \overline{1, m}$

It's most general form in which:

1. Regions can intersect
2. Regions can have arbitarly form,
3. It does not need that regions cover all function domain

5. Decision Trees, regression/Score

Usual standard for Loss in regression is quadratic function of discrepancy.

And used loss in CART is also quadratic loss.

Score criteria – least squares scaled by N

$$s(\{c_m, R_m\}) = N \cdot R(F) = N \cdot \frac{1}{N} \sum_{i=1}^{N} (y_i - \sum_{m=1}^{M} c_m I(x_i \epsilon R_m))^2$$

Optimization problem

$$\{c_m, R_m\} = argmin_{\{c_m, R_m\}} s(\{c_m, R_m\}) = argmin_{\{c_m, R_m\}} \sum_{i=1}^{N} \left(y_i - \sum_{m=1}^{M} c_m I(x_i \epsilon R_m) \right)^2$$

Welcome to real problem:

Our optimization problem has objective which:

1) non-convex
2) non-differentiable
3) non-continious

Search strategy – is under big question

5. Decision Trees, regression/ Search Strategy(opt.problem solving)

Let's make relaxations to make problem feasible to solve

Assumption #1

1. $\left(R_i \cap R_j\right)_{i \neq j} = 0$ i.e. regions are disjoint
2. Regions cover space of all possible values.

After this assumption:

1. $F_c(x) = \sum_{m=1}^{M} c_m I(x \epsilon R_m) = c_m \ (where\ m\ is\ such\ that x \epsilon R_m)$

2. If regions are known then optimal c_k equal to arithmetic mean of y_i in each k region in case of fixed sets of region. (Prove in [3], 8.2)

5. Decision Trees, regression/ Search Strategy(opt.problem solving)

Optimization Problem

$$\{c_m, R_m\} = argmin_{\{c_m, R_m\}} \sum_{i=1}^{N} \left(y_i - \sum_{m=1}^{M} c_m I(x_i \epsilon R_m) \right)^2$$

For fixed $\boldsymbol{R_k}$ satisfy **Assumption #1**

$$\boldsymbol{c_k} = \frac{\sum_{i=1}^{N} \boldsymbol{y_i} \boldsymbol{I}(\boldsymbol{x_i} \boldsymbol{\epsilon} \boldsymbol{R_k})}{\sum_{i=1}^{M} \boldsymbol{I}(\boldsymbol{x_i} \boldsymbol{\epsilon} \boldsymbol{R_k})}$$

But we still do not know the regions! And unfortunately for arbitrarily regions it's still **hard combinatorial problem**. And we need extra assumption.

5. Decision Trees, regression/ Search Strategy(opt.problem solving)

Assumption #2: $R_i = S_{i1} x S_{i2} \ldots x S_{im}$

Regions are Cartesian product of the set S_{ik}.

Each S_{ik} is subset of possible x_k values in special form:

(e.g. interval for interval scale or ordinal x_k)

(e.g. subset if x_k is categorical)

***Consequences*:**

1. $I(x \epsilon R_m) = \prod_i \ I(x_i \epsilon S_{mi})$
2. $F_c(x) = \sum_{m=1}^{M} c_m I(x \epsilon R_m) = \sum_{m=1}^{M} c_m \left(\prod_i \ I(x_i \epsilon S_{mi}) \right)$
3. $\sum_{m=1}^{M} \left(\prod_i \ I(x_i \epsilon S_{mi}) \right) = 1$

5. Decision Trees, regression/ Search Strategy(opt.problem solving)

- So we want to partition function domain with regions
- Regions has specific (hyper)rectangular form
- In each region we predict constant value

We can think about extra things:

- More complicated regions with less count OR more rect. like regions

"*…As usual everything in this business it depends.. "* – *J.Friedman*

- For regression trees we predict one single value as a "response", it's incomplete information.

"*…Very few people in this world try to analyse it more precisely.. "* – *J.Friedman*

- In regression tree we will try to collect samples with similar y_i even they can belong to different distributions.

5. Decision Trees, regression/ Search Strategy(opt.problem solving)

$$\{c_m, R_m\} = argmin_{\{c_m, R_m\}} \sum_{i=1}^{N} \left(y_i - \sum_{m=1}^{M} c_m \left(\prod_j I(x_i \epsilon S_{mj}) \right) \right)^2$$

$$R_i = S_{i1} x S_{i2} \dots x S_{im}$$

Drama. We still can not solve it. It is still hard combinatorial problem. This days we can solve this exactly for 7 regions, but it's still laughable.

Common heuristic approach is ***Recursive Partitioning***.
It's rediscovered in various areas. It's greedy, iterative algorithm.
And CART use it too.

5. Decision Trees, regression/ Final Search Strategy

- Step-1: Start with entire region $R_1 = S$ and $F_1(x) = \bar{y}$
- Step-2: At M-th iteration we have $F_m = \sum_{m=1}^{M} c_m I(x \epsilon R_m)$
- Step-3: Choose (parent) region to partition into *group* of **two** daughter regions
- Step-4: Replace parent with two daughters
- Step-5: Now we can update counter of total regions M +=1 and goto step-2

5. Decision Trees, regression/ Why partially constant functions called a tree?

In principle – sequence of splits can be represented as a tree where:

- Terminal nodes represents final regions associated with the model
- Internal nodes represents - splits

6. Decision Trees, regression/ 3 main questions

Problem: It's very crude algorithm, but it gives computational feasible strategy. Three major questions for algorithm:

- Q1: Which regions select to split within M regions?
- Q2: In which variable within a region perform a split ?
- Q3: When stop splitting ?

Split region as opt.problem

In regression tree we consider splitting region R_m into R_l and R_r

- $R_m = R_l \cup R_r$
- $R_l \cap R_r = 0$

And we find split which maximize the following objective:

$max.improve$

$$improve = N\left(\frac{1}{N}\left(\sum_{i=1}^{N}\left[y_i - F_{c_{before\ split}}(x_i)\right]^2\right) - \frac{1}{N}\left(\sum_{i=1}^{N}\left[y_i - F_{c_{after\ split}}(x_i)\right]^2\right)\right)$$

(Implicit constraint $improve \geq 0$*)*

It will give us maximum decrease (via greedy step) for **risk of entire tree scaled by N**

$$R = N\frac{1}{N}\sum_{i=1}^{N}\left(y_i - \sum_{m=1}^{M} c_m\left(\prod_j I\left(x_i \epsilon S_{mj}\right)\right)\right)^2$$

And if $improve = 0$ **it will just stops algorithm due to it limitations**

Split region as opt.problem

We consider splitting $R_m = R_l \cup R_r$

But for points which are not belong to R_m:

1. Geometry of regions in which they lie will not change
2. Train set/Sample is fixed and is not changing during learning/fitting
3. Optimal c_z equal to mean value of y_i in the region z.

Consequence
For evaluate improvement for split region **m** we can consider only R_m and points belong to it and just forget about other regions.

And so:

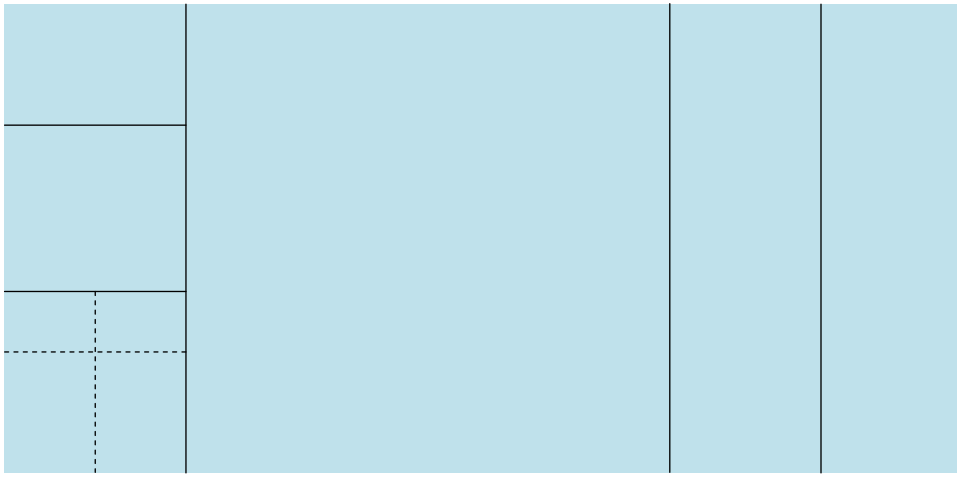
$$improve = \left(\sum_{i:x_i \in R_m} [y_i - F_{c_m}(x_i)]^2\right) - \left(\sum_{i:x_i \in R_l} [y_i - F_{c_l}(x_i)]^2\right) - \left(\sum_{i:x_i \in R_r} [y_i - F_{c_r}(x_i)]^2\right) \geq 0$$

Q1:Which regions select to split within M regions?

Answer:

Find best split for each region and then find the "best from the best".

What is good is that we should not re-do all this computation at each iteration.



Q2: In which variable within a region perform a split ?

Answer: Try all variables and find best split for each one of them.

Ok, how find best split point for one single variable?

Q2: *How find best split point for one single variables/ How evaluate quality of split ?*

$$improve = \left(\sum_{i:x_i \in R_m} [y_i - F_{c_m}(x_i)]^2 \right) - \left(\sum_{i:x_i \in R_l} [y_i - F_{c_l}(x_i)]^2 \right) - \left(\sum_{i:x_i \in R_r} [y_i - F_{c_r}(x_i)]^2 \right) \geq 0$$

With derivations [3], 7.4.2 it can be proved that this can be evaluated as:

$$improve = \frac{n_l n_r}{n} (\bar{y}_l - \bar{y}_r)^2$$

- n_l – number of observation in left daughter of region Rm
- n_r – number of observation in right daughter of region Rm
- n – number of observations in Rm.
- $\bar{y}_l, \bar{y}_r$ are means of outcome variable in the left and right regions
- F_{c_m} - approximation for region **m,** optimally constructed approximation for left/right regions: F_{c_l}, F_{c_r}

(I think this can be found in [1] or [2] – I just did not look into it and it have been proved as homework)
What is good about it:

- If move split point to the right or to the left(or from left to right) all quantities can be update computation easily.
- It's like rank-one update formula for ordinary-least-squares

Q2: *How find best split point for one single variables?*

- **ordinal variable** – sort and exhaustive search for maximum *improve* via all possible split points in time proportional to $\sim \mathrm{card}(type)$
  (Russian equivalent for card here - мощность множества)
- **Interval scale** – even possible split points are infinite, our data is not, so we just do sort and exhaustive search of split point
- Nobody don't know where to insert splitting point within *two neighboors* and people insert it into middle in case of interval scale
- Out *improve* quantity is insensitive where precisely insert split point between two neighboors

Q2: *How find best split point for one single variable which is catgorical*

- Categorical variable is the most **tricky**.
  - **There are no order relation.**
  - **If consider subset** then here are $2^{\mathrm{card}(type)}$ possible splits.
- Step-0: We have observation which are in current region in which we consider splitting.
- Step-1: For each category of the variable we evaluate mean $\overline{y_c}$ value. This computation effort of this step is proportional to O(n).
- Step-2: Give each class score which is equal to this mean
- Step-3: Sort them and consider categories in this order
- Step-4: Do similar thing that we do for ordinal variables via **consider only $\mathbf{card(type)}$ splits, not $\mathbf{2^{card(type)}}$**

"There is a trick when it have been invented it was assumed as heuristic, but in future it have been proved that this is absolutely correct and is exact. Prove is very-very complicated" – J.Friedman

Q2(Extra): *How find best split point for one single variable which is circular*

How handle circular variables

- *Step-1*: If variable used for splitting first time then make split – find optimal two split points and optimize over both

- *Step-2*: Other splits on this variable use usual splits in arc segment of the circles

"I don't know any implementation which implemented it" – J.Friedman

Q3: When stop splitting ?

Bad idea:
Stop when number of observation in region is less then threshold.

Reason:
It depends on situation:
- We can start to overfitt train data
- Or we can not fit perfectly

More good idea:

Next split is not worthwile.
Best split is the best split which can be archieved via one look-ahead.
Unfortunately for one reason it's also not so good idea because

"It often happens that a single split is not worthwhile by itself, but it opens possibility for further splitting" – J.Friedman

Q3: When stop splitting ? Final idea

- Complete Mean Square Error(MSE) is equal
  $\hat{e} = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \widehat{y_{m:m\ is\ such\ that\ x_i \in R_m}}\right)^2$
- Partial MSE is equal to
  $\widehat{e_m} = \frac{1}{N}\sum_{i:x_i \in R_m}(y_i - \widehat{y_m})^2$
- $\widehat{e_m}$ is contribution to complete MSE from the R_m
- In both equations N is number of all observation
- $\hat{e} = \sum_{i=1}^{M} \widehat{e_i}$

Q3: When stop splitting ? Final idea

- Let's define Improvement of apply splitting is equal to $\hat{I} = \widehat{e_m} - \widehat{e_r} - \widehat{e_l} \geq 0$
- If $\hat{I} \geq k$ then accept split.
- $\boldsymbol{k}$ is cost for perform splitting. k is also known as complexity parameter (**cp** parameter in *rpart*)
- We introduce quanity

$$\widehat{c_m} = \begin{cases} \widehat{e_m} + k, m \text{ is terminal} \\ \widehat{c_l} + \widehat{c_r} = \sum_{m' \text{is terminal descendant of } \boldsymbol{m}} c_{m'} \end{cases}$$

- In fact we accept split

$$\widehat{e_m} - \widehat{e_r} - \widehat{e_l} \geq k <=> \widehat{e_m} + k \geq \widehat{e_r} + \widehat{e_l} + k + k <=> \widehat{c_m} \geq \widehat{c_l} + \widehat{c_r}$$

CART. Final algorithm

- **Step-1**: We do **top down splitting** as far as possible until we receive only one single observation in the terminal node (or all y_i within a region is the same)
- Such splitting is not real "look ahead". We find optimal split point and continue build of tree from this point **without backtrack greedily**.
- **Step-2**: We apply **bottom up rule** in inverse order of depth and perform collapsing.
  - If $\widehat{c_m} \leq \widehat{c_{ml}} + \widehat{c_{mr}}$ then make **m** as a terminal node and collapse split
  - If $\widehat{c_m} \geq \widehat{c_{ml}} + \widehat{c_{mr}}$ then accept split and set $\widehat{c_m} = \widehat{c_{ml}} + \widehat{c_{mr}}$
- Collapsing can happened in the level before leafs, and also somewhere in the middle of the tree.

CART. How find cp.

- All meta-parameters in Machine Learning can be found via cross-validation

- Cross-validation is mechanism which
  - Allow to approximately estimate the performance of the actual predicting. There is no magic in it.
  - In fact when cross-validation is doing we should trace even after local minimum on cross-validated point where we archived good result

Summary.Advantages of Trees

1. Resistance for many irrelevant variables (e.g. SVM is not irrelevant)
2. Allow to visualize how function is evaluated in high-dim. Space
3. Relatively fast. Build decision tree cost $\sim n \cdot Nlog(N)$ (worst N^2)
4. Where: (N-number of observations, n – number of variables)
5. Support a lot type of variables: *numeric, binary, categorical, ordinal*
6. Can handle missing values during inference in light way
7. Invariant for monotone transform of predictor function domain
8. Computation algorithm is invariant under variable *scaling*.
9. Immunity to outliers due to partitioning of domain
10. Interpretability in terms of conjuction rule.
11. Give full explanation how it makes prediction
12. Few tunable parameters

Summary. Disdvantages of Trees

1. Very crude model, especially because in nature more things are continuous, not discontinuous. (Or at least there are both)

1. There is a special notion of interactions. Empirical observation - 3 variable interaction covers at most all phenomens.
    1. ***Main effects*** – is when approximation of phenomena is given by the sum of single scalar variable functions
    2. ***Two variables interaction*** – when there is main effects and with also two variables coupling
    3. ***Three variables interaction*** – similar as for two variable interaction but with 3 variable coupling (ANOVA)

2. Trees by their nature are forced to build high-order interaction

3. Data fragmentation. During split's there are less and less data in daughter regions.

4. One more problem is that **trees have big variance**.
    1. The possible error in upper part of decision tree propagates down to the terminal nodes.
    2. The errors are not accumulated (or averaged), but in fact errors is cascaded/multiplied
    3. Even minor change split in the root will dramatically change tree structure

5. How to fix variance (within a room of this model):
    1. Consider more samples
    2. Consider less function space (number of regions)

Appendix A. Missing data

- People don't provide data because:
  - They are missing
  - They are lost
  - Person who give data have real reasons to not give it to you this specific (features) data

Appendix A. Missing data. Idea is exploit localized region

This problem is mitigated by "*surrogate split*" in context of decision trees.

Algorithm:

1. Create split point for x_s based on not missing examples for observation within this region
2. Score observation from left as ***0*** and observation from right as ***1*** in **current region**
1. Goto all observation with not missed values for other variables in current localized region
2. Look how another variables can make good prediction for **0/1**
3. Now beside *primary split* (x_{31}, S_{31}) we store bunch of surrogate splits
4. If variable for surrogate split is absent too then next best surrogate split is using. If primary and surrogate splits are missing for observation then we go left/right at random based on proportion left/right from train.

Appendix B. Variable importance

Several possible ways to define importance

1. What variables have been used during prediction. Via following the path from top to bottom all variables that have been occurred in inference/evaluation are *equally important*.
2. Score/weight (1) by "*improvements*" evaluated during training.
3. Variable importance is measure how **strong this variable** can be used to build predictor with only this variable and no others.
4. Another way measure variable importance is **measure how terrible will predictor be if no use specific variable**