Topic:

"Story about decision trees based on lectures from one professor" (Part II/III)

Konstantin Burlachenko

-kburlachenko@nvidia.com

-bruziuz@stanford.edu

-burlachenkok@gmail.com

References

[1] First part of presentation
https://github.com/burlachenkok/presentations_bruziuz/tree/master/decision_trees/mpti

[2] Classification And Regression Trees, 1983 (Brieman, Friedman, Olshey, Stone) – CART
https://www.amazon.com/Classification-Regression-Wadsworth-Statistics-Probability/dp/0412048418

[3] Book: The Elements of Statistical Learning
(Friedman, Tibshirani, Hastie)
https://web.stanford.edu/~hastie/Papers/ESLII.pdf

[4] News: Bradley Efron has won the 80K USD Prize in Statistics for his 1970s work
https://www.nature.com/articles/d41586-018-07395-w?fbclid=IwAR3EbV3LbBQmls4KarxRzblvonn6HUxiC4IqRGaaXPY2XRsbOXtSPHq03pk

[5] [cvxbook] Convex Optimization (S.Boyd, L.Vandenberghe)
http://stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf

[6] My notes: What is cross-validation and some hints about it
https://sites.google.com/site/burlachenkok/articles/what-is-cross-validation-and-some-hints-about-it

[7] Example of Random Forest usage.
Real-Time Human Pose Recognition in Parts from a Single Depth Image, 2011
https://www.microsoft.com/en-us/research/publication/real-time-human-pose-recognition-in-parts-from-a-single-depth-image/?from=http%3A%2F%2Fresearch.microsoft.com%2Fpubs%2F145347%2Fbodypartrecognition.pdf

[8] Some way to interpretate black-box models
https://sites.google.com/site/burlachenkok/some-ways-to-intepretate-black-box-models

Plan

1. Reminder about Decision tree for Regression
2. Extremely short note about decision tree for classification
3. Problem with regression trees and what todo
4. Ensemble of trees. Why such thing matter.
5. Bagging
6. Random Forest
7. Bagging and Random Forest. Advantages
8. Slide about cross-validation (or x-validation)
9. Boosting
    1. Motivation
    2. .....
    3. .....
    4. .....
    5. .....
    6. .....
    7. .....
    8. .....

CART Decision Tree for regression. General

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

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

After several relaxations

- We come to this formulation for building piecewise constant function.
- Here in this formulation and more broadly in all Machine Learning people think that they know precisely (x_i, y_i), even it is not true (as have been mentioned in [1] part).
- People in ML do not use any *robust schemas* to handle uncertainty in (x_i, y_i)

Drama

Even in such formalization we can not solve it exactly and efficiently - It is still hard combinatorial problem. Common heuristic is ***Recursive Partitioning***.

CART Decision Tree for regression. Split.

In each iteration of refinement 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

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

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

CART Decision Tree for classification (easy upgrade of model structure)

- Schema can be upgrade for inference

$$F_c(x) = \sum_{m=1}^{M} c_m \left(\prod_j \; I\left(x_j \epsilon S_{mj}\right)\right)$$

Where:

1. c_m now is not scalar from $\mathbb{R}$ but element of finite set $C = \{class_1, class_2, \dots, class_K\}$
2. x is input vector for predictor and x_j is it's components
3. $R_m = S_{m1} \times S_{m2} \times \cdots \times S_{mn}$ Region is presented as cartesian product of simple sets

CART Decision Tree for classification (various problems in search strategy)

Assume we leave score criteria as it was for regression

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

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

Drama:

Greedy one look-ahead does not work at all in such circumstances with improvement .

To upgrade CART(Classification And Regression Trees) model to allow make classification various steps should be made.

CART Decision Tree for classification. (Only general picture how this model handle classification)

- Construct predictor which estimate (interval-scale) probability of each class for random variable $Y|X$ instead of estimating value of $Y|X$ directly.

  This trick allow to remove this strange categorical variables and come to interval scale

- We solve K-response regression problem with affine constraint that sum of all response is equal to 1 which allow to construct approximation of such probabilities

- Mathematical manipulation should be done

- Conclusion what is important during splitting for it is

  "Probability be in region" x "Diversity of the regions"

  Where Diversity is estimated by *Gini index of diversity*

- Gini index of diversity $G(p) = 1 - p^T p$ *on probability simplex*
  - *It archives maximum when* $p_i = \frac{1}{K}$
  - *It archives minimum when* p *probablity mas function is discrete delta*

CART Decision Tree for classification. (Only general picture how it is doing)

After various reformulation we can come to

- We can solve regression problem like we do in CART for regression, but we Find split which **maximize improve**:
  $$improve = P(R_m)Gini(R_m) - P(R_l)Gini(R_l) - P(R_r)Gini(R_r)$$

- Also because $P(R_m) = P(R_l) + P(R_r)$ we in fact can more easily to compute:
  $$improve = P(R_m)H(R_m) - P(R_l)H(R_l) - P(R_r)H(R_r)$$

- Quanity $\mathrm{H(p)} = -\mathrm{p}^{\mathrm{T}}\mathrm{p}$ sometimes call **"second order entropy"**

- Author of C.4.5 used not H(p) but use usual entropy ($-p_i \log(p_i)$) instead

- In leafs regions probability of each class is just computed as
  $$p_c = \frac{\#number\ of\ (x_i,y_i)\ where\ y_i=c}{\#total\ number\ of\ (x_i,y_i) from\ region}$$

To cover it in glorry detail - a separate presentation needed

Problem with decision trees for regression

Problem is that **trees have big variance** because:

1. Very rapid data fragmentation. During split's there are less and less data in daughter regions.
2. The possible error in upper part of decision tree propagates down to the terminal nodes.
3. The errors are not accumulated (or averaged), but in fact errors is cascaded/multiplied
4. Even minor change split in the root will dramatically change tree structure

What todo if we do not leave room of nice advantages of the decision trees?

- Live with the problem
- Fix up trees. Recent research is:
  1. **Bagging ,1996**
  2. **Boosting, 1996**
  3. **MARS (Multiple Adaptive Regression Splines) 1989.** (not so popular then 1,2 right now)

Ensemble of trees. Why such thing matter

First view of ensemble of trees

- Ensemble of tree can be viewed as linear combination of tees
- Trees is piecewise constant function with special form of regions
- Linear combination of piecewise constant function is piecewise constant function

But why does it matter

1. In ensemble of trees there are more pieces then in single tree
2. Pieces can overlap
3. Any function can be approximated by piecewise constant functions if there are:
    1. A lot of regions to split domain
    2. A lot of data to fit each constant value in each region

Bagging. History and goal

- Bagging was invented by Leo Brieman in 1996. Reference in the book **[3], ch 8.7**
- Goal is improve behaviour of unstable predictive schemas like:
  - Neural Nets
  - Decision Trees

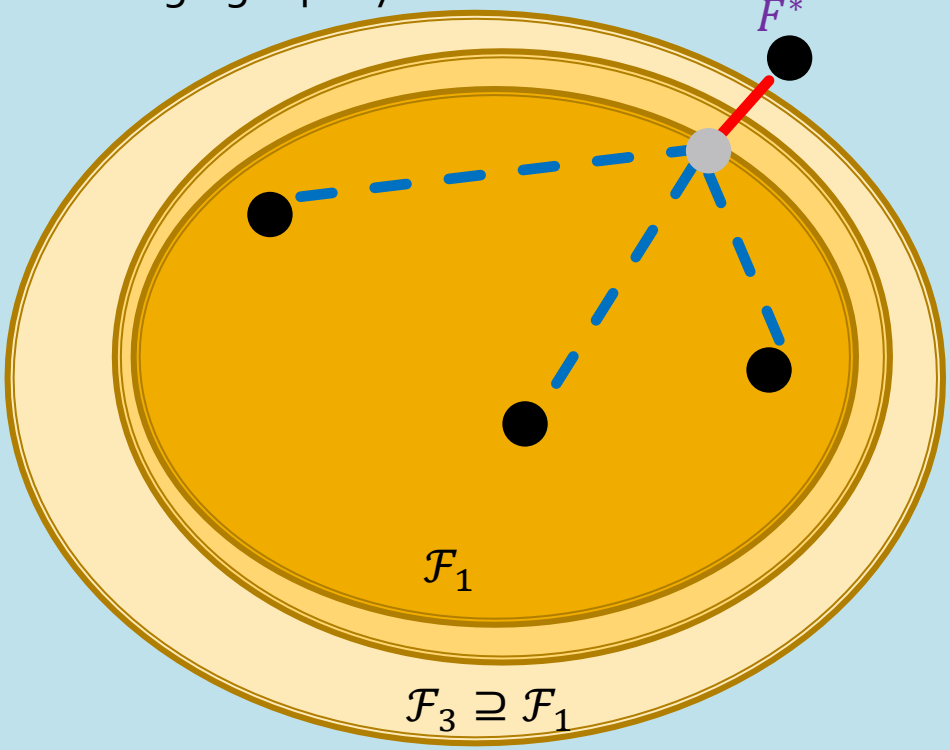
Unstable means the following.
We have optimization problem for empirical Loss minimization:

$$\hat{F} = argmin_{F(x)\in\mathcal{F}} \left(\frac{1}{N}\sum_{i=1}^{N} L(y_i, F(x_i))\right)$$

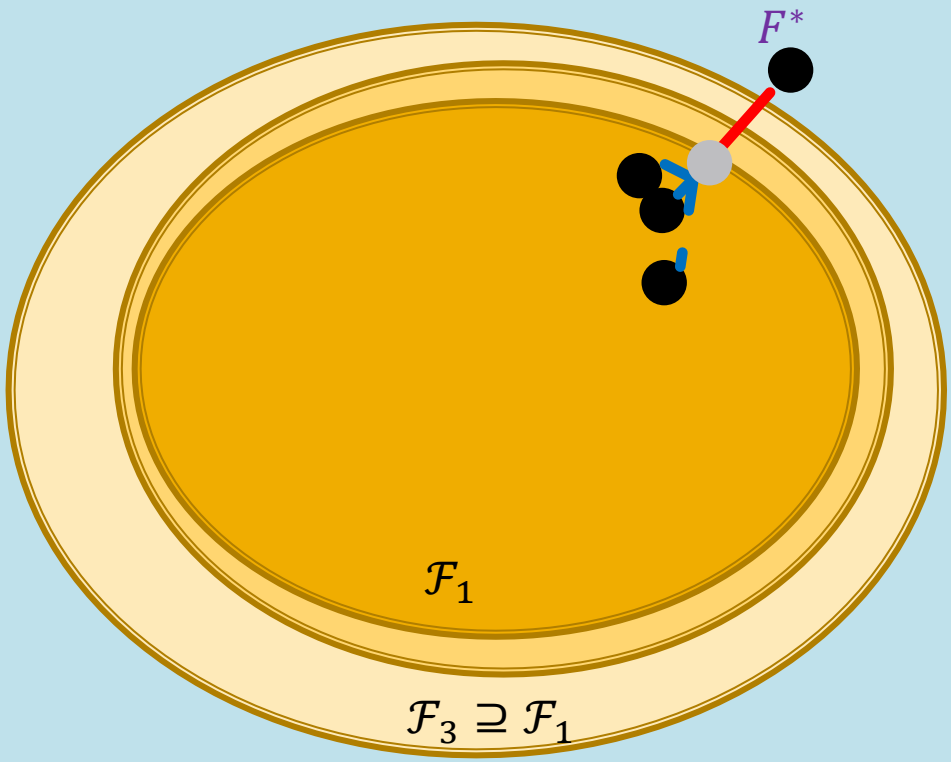
But if we will perturb "train set" (X, Y) then it will lead to dramatically change of $\hat{F}$.

Bagging. Illustration of unstable procedure behaviour when function class is big.

Distributions of solutions is
"very wild as changing rapidly" as we draw different "train set" from population



Bagging. Illustration of stable procedure behaviour when function class is big.



14

Bagging. Several quotes.

"The simplicity how Leo figure out how to fix it is awesome by it's simplicity" – J.H. Friedman

"Leo called it's bagging because it means bootstrap aggregation" – J.H.Friedman

Bagging algorithm Step 1/3

Step-1: Perturb original train data T in some way and get perturbed train set T_b

"In fact it doesn't matter a lot how to make change and there are many ways todo it" – J.Friedman.

One fancy technic is "bootstrap sample".

We have set T contains N observations. We sample randomly observation from this train data and each iteration we sample *observation* we back sample to train data.

Bootstrap technic was invented by *Brad Efford* for other things.

By the way **12NOV2018** he received 80K USD price International Prize in Statistics for his 1970s work [4]

Jerome H.Friedman:
"Another technic which Leo tried add random noise to Y. And it works pretty well too. In any case idea is make perturbation a bit, but not too much."

Bagging algorithm Step 2/3

Step-2:

Apply original procedure with pulling "best" function from our function family via search strategy:

$$\hat{F}_b = argmin_{F(x)\in\mathcal{F}} \left(\frac{1}{N} \sum_{i=1}^{N} L(y_i, F(x_i)) \right), (x_i, y_i) \in T_b$$

Bagging algorithm Step 3/3

Step-3:

Repeat step-1, step-2 "*B*" times. Average behavior of "B" models and give final model as average of B models

$$\hat{F} = \frac{1}{B}\left(\sum_{b=1}^{B} \hat{F}_b\,(x)\right)$$

Bagging. Some observations

- Bagging doesn't do nothing with bias (almost always). If we consider function space which is closed at least to "sum/scaling".

⇒ So particularly for tree's "*this silly thing*" do nothing with bias error.

- Also what is averaging is **ouput** and **it's not the structure** of the trees.

- *"But why this thing can do something? How even it's possible! " – J.Friedman*

Bagging . Why this thing is working?

Deep answer lie in area of math optimization and one aspect of difference between **convex** and **non-convex** math optimization.

Bagging . About perturbed convex optimization problem

Let's talk a bit on convex optimization problem and it's perturbations

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq u_i \\ & h_i(x) = v_i \end{array}$$

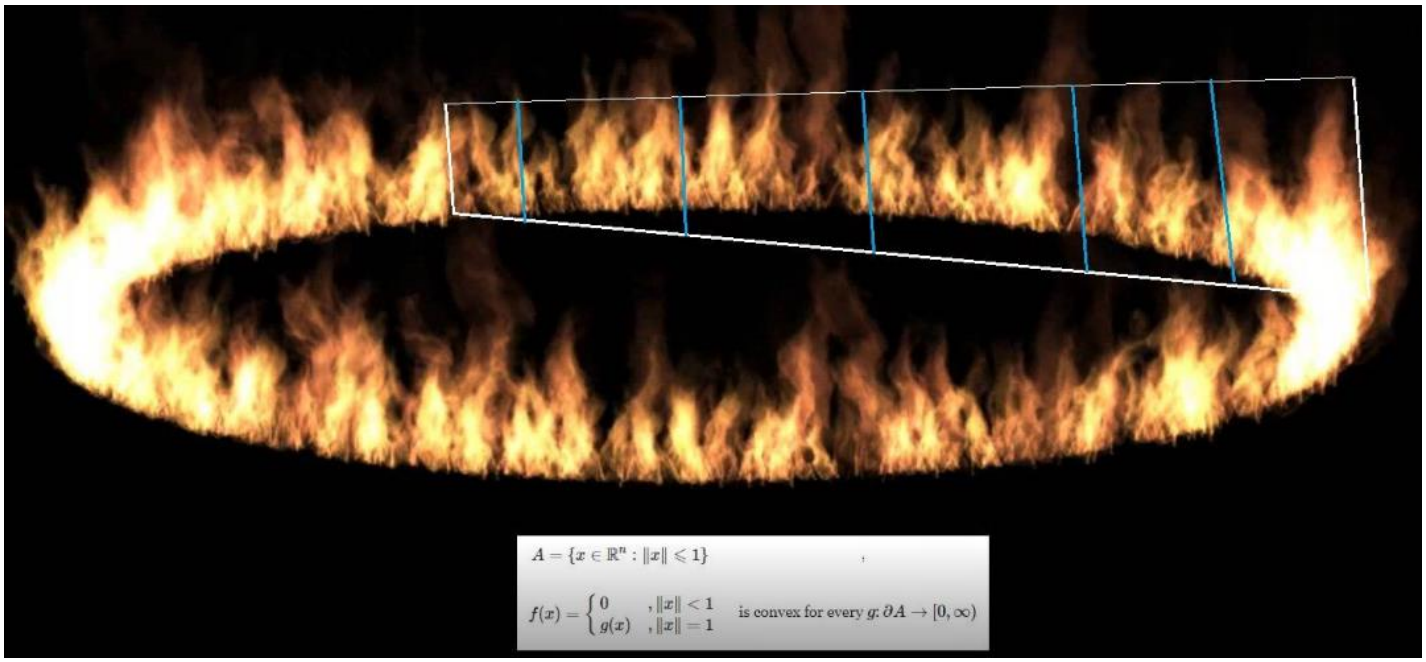
- **Optimal value $\boldsymbol{p}^*$** – is infimum of objective function attained in feasible set and intersected with the domain of optimization problem [cvxbook], p.127
- **Optimal value of the perturbed convex optimization problem** during perturbation of right hand side equality or inequality **is another convex function** of the parameters of this perturbations
- If optimal value of perturbed problem is differentiable then for the perturbed optimal value as a function of perturbations $\boldsymbol{p}^*(\boldsymbol{u}, \boldsymbol{v})$ we have: $\frac{\partial p^*(0,0)}{\partial u_i} = -\lambda_i^*$ and $\frac{\partial p^*(0,0)}{\partial v_i} = -v_i^*$

  => So in this case if perturbation are not big too much then optimal value roughly speaking is not changing too much

[cvxbook], 5.6. Perturbation Analysis, p.250,p.234-236

Bagging . About convex functions.

Convex function are continuous in relative interior of it's domain. So mostly they are continuous even some strange situations can happend on boundary:



https://www.facebook.com/photo.php?fbid=664179037304832&set=a.164584463930961&type=3&theater

Bagging. Will bagging work for problem which are reduced to convex optimization?

- **No**

"Bagging does not help for procedures which are themselves are convex optimization problem" – Jerome H. Friedman.

"Whole point of ensemble you combine different decision trees… ensemble methods in context of convex optimization are boring…" – S.Boyd, 58:40, https://www.youtube.com/watch?v=wqy-og_7SLs

Bagging. Ideas why bagging helps for approximation schemas reduced to non-convex optimization

1. Let's start. We have multiple local minimums in our objective.

2. If use local iterative strategy then obtained optimal point, depends on our start.
   (By the way one more fun moment with S.Boyd https://youtu.be/wqy-og_7SLs?t=2953, 49:12
   "*If you have non-convex problems and we will use solvers that you mentioned. Stephen: It depends, but the most accurate technical statements is the following.* ***Something happens***")
3. So change in starting point will lead to different solution.
4. We don't change starting point in this strategy. What we're doing – we perturb/shuffle a bit/change dataset. Which is the same as change start point if think about it.

We hope:

1. That there is global picture of convex function with several ripples.
2. Near global minimum there are several other local minimums, but they are clustered near each other
3. Sampling technic will more probably give us predictors with minimums which are near global minimums
4. Another thing that for convex function we have $f(E[w]) \leq E[f(w)]$.
   So for convex function averaging in model obtained by averaging parameters is better then average of the models

Empirically:

This technic from 1996 upgrade Decision Trees into extremely powerfull model.

Bagging. Nice decoupling.

- Via deep of decision trees – we control bias.
- Via bagging technic – we control variance .
- If to be completely honest we can bag anything
- **When bagging works**
  - Trees are not highly correlated
  - Each tree should be enough strong to predict something
- But Bagging as "average technic" works with improve quality in average, no for particular tree.
- *"It still can be a case then a single tree works better"* – J.Friedman

Random Forest as extreme case of Bagging

- Random Forest is most popular form of bagging
  - We create deepest Decision Trees without any pruning (cp=0)
  - The height of decision tree is only limited by the size of train samples
  - Wrap-around this procedure via bagging
- Nice thing about Random Forest – each tree can be compute completely in parallel.
- Each Decision Tree is building completely independent
- Example of usage. **Kinect** in human pose recognition use Random Forest [7]

26

Random Forest. Conclusion.

- In Random forest it's possible append extra variation of the trees via build tree based on random subset of variables.
- J.Friedman said that Leo didn't give any recommendation of the number of variables, but via consideration of J.Friedman selecting of variable **is just a strange thing**.
- For random forest it should be 2 meta-parameters:
  Size of the sample for bootstap.
  Smaller boostrap sample – more randomness in the tree.
  Bigger boostrap sample – less randomness in the tree.
  Fraction of selected variables.
- Unfortunately in standard implementations there is no necessary such thing have been implemented both.

Bagging and Random Forest. Advantages

- They share all nice properties of Decision Trees from [1] **exclude 1** interpretability
- Even If you have 1000 or 10 or **even 5** trees it's very hard to understand what is going on under the hood (or at least not so easy)
- But main competitor which is *Neural Nets* – is not interpretable too (or at least it's not so easy)

(If you're interesting in interpretability here a short note about several schemas to interpret black-box models in AI/ML [8])

Model selection: slide about cross-validation

- Several times this word come into play I need to say at least something.
- Let's me describe the simplest way todo it.
  1. It's schema when you have opt.problem with variables **(a, λ)** and you split you data which you use to fit/extrapolate into two **sets** – train and **test**
  2. You solve opt.problem on a based on **train** data for some fixed λ
  3. You check how problem behaves on data called **test**
  4. Do if for several λ

ML community do not state that they want explicitly so decision which one λ to choose is up software engineer.
(usually smaller is better).

In terms of math optimization some variation of such strategy can be formalized as the following. We have two sets Set_1 and Set_2 with data . We can call it **train** and **test,** but It's only a name) and we solve the following opt.problem:

$$min. \sup_{S \in \{Set_1, Set_2\}} \left(R_S\left(F(a,\lambda)\right)\right)$$

variables: a, λ (optimization community or when optimization based model arised in ML/AI)
parameters: a, λ (machine learning community)

In such formulation it's identical to **robust worst case optimization**.
If λ is something wild enough then from algorithm above step #4 can be repeated several times

Also here my note "*What is cross-validation and some hints about it*":
https://sites.google.com/site/burlachenkok/articles/what-is-cross-validation-and-some-hints-about-it

Boosting. Only motivation

"Motivation as I see it's right now is completely different how it was historically developed. It's not in our book [3], but I think it's just better way to understand from Optimization perspective" - J.H.Friedman, May2018.

Motivation: build something like random forest

But now similarity stops…

To be continue in the middle of February,2018…

Merry Christmas and a happy new year!