

ML for the industry

Part 1

MLSS 2016 – Cádiz

Nicolas Le Roux

Criteo

Why such a class?

- Companies are an ever growing opportunity for ML researchers
- Academics know about the publications of these companies
- ...but not about the less academically-visible research

A new zoology of problems

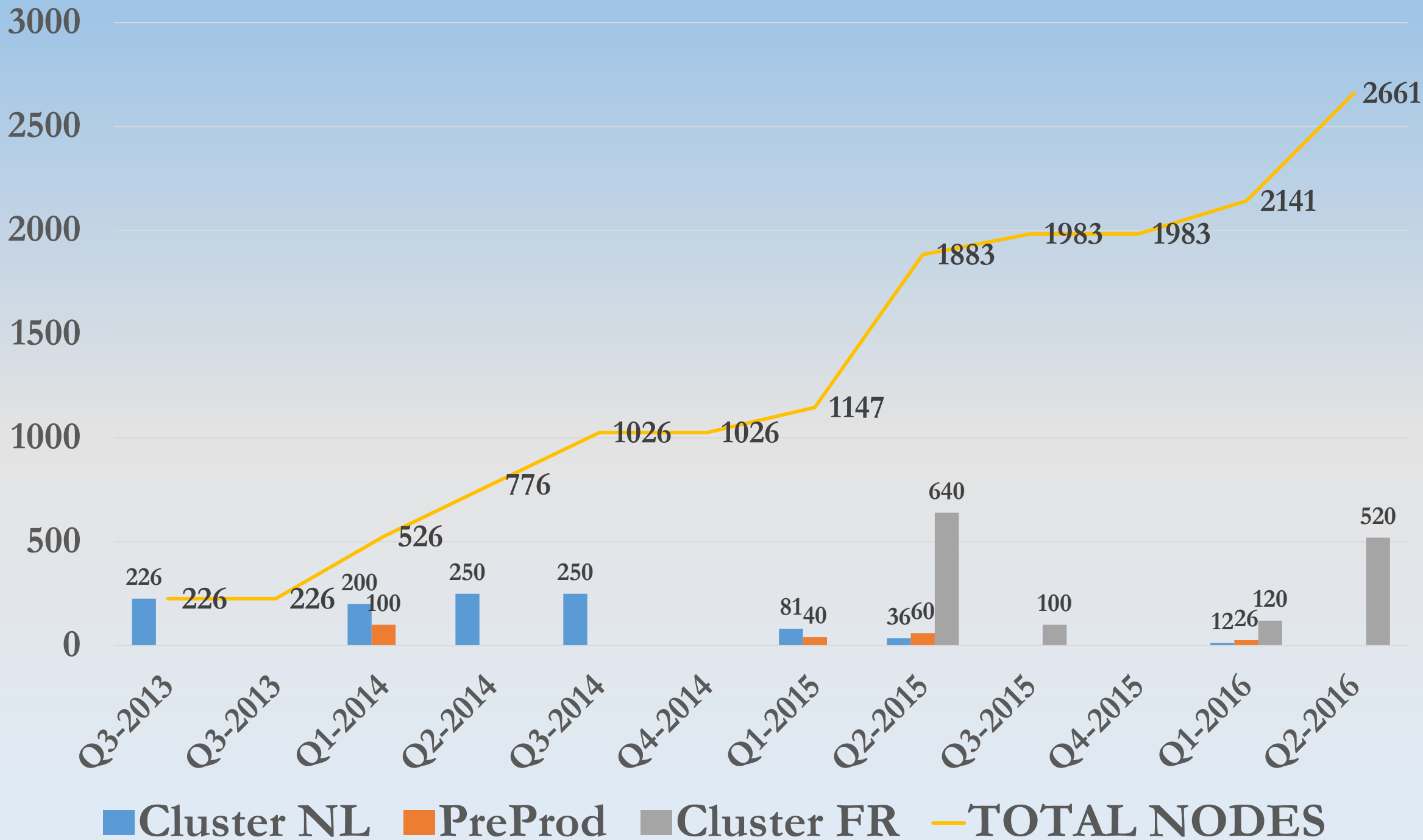
- Most academic literature is about predictive performance
- What about:
 - Optimisation of decision-making?
 - Increasing operational efficiency?
 - Predictive performance under operational constraints?

The 3 stages of the academia → industry move

1. I will use model X which will greatly improve the results (enthusiasm)
2. No new model is useful, this is pointless (disillusionment)
3. So many open questions, I do not know where to start (acceptance)

Criteo – an example amongst many

- We buy advertising spaces on websites
- We display ads for our partners
- We get paid if the user clicks on the ad



Retargeting – an example


LIVRAISON GRATUITE
des 15€ d'achat

RETOUR & ECHANGE
GRATUITS

EXPÉDITION
100% des articles

OFFRES FIDÉLITÉ
1 commande = 1 cadeau-cadeau

HE LOUN | Collection Automne-Hiver 2014 / Volcom / Pulls / Gilets



VOLCOM
Gilet zippe Lived

64€ / 32€
Vous économisez : 32€


EN STOCK
LIVRAISON ET RETOUR GRATUITS

64 pts = 6.40€ OFFERTS

ACCUEIL | DEPOSER UNE ANNONCE | OFFRES | DEMANDES | MES ANNONCES | BOUTIQUES | MON COMPTE | AIDE

Accueil > Seine-Saint-Denis > Jeux & Jouets > Poney à bascule musical : billy joe poney basculo

Poney à bascule musical : billy joe poney basculo




CONTACTER VVA972

Envoyer un email

GÉRER VOTRE ANNONCE

Modifier | Supprimer

Mettre en avant | Remonter en tête de liste



MONSHOWROOM.com
LIVRAISON ET RETOUR GRATUITS

In practice

1. A user lands on a webpage
2. The website Criteo and its competitors
3. It is an auction: each competitor tells how much it bids
4. The highest bidder wins the right to display an ad

Details of the auction

- Real-time bidding (RTB)
- Second-price auction: the winner pays the second highest price
- Optimal strategy: bid the expected gain
- Expected gain = price per click (CPC) * probability of click (CTR)

What to do once we win the display?

- We are now directly in contact with the website
- Choose the best products
- Choose the color, the font and the layout

Identified ML problems

- Prediction problem: click/no click
- Recommendation problem: find the top products

What is the input?

- The list of data we can collect about the user and the context
- Time since last visit, current URL, etc.
- There is potentially no limit to the number of variables in X

Choosing a model class

- Response time is critical
- There is little signal to predict clicks: we need to add features often
- Solution: a logistic regression - $\text{pCTR} = \sigma(w^T x)$

A major difference

Structured data

- Lots of info in the data
- High predictability
- Highly structured info

Hierarchical models

Unstructured data

- Poor predictability
- Signal dominated by noise
- Highly unstructured info

Linear models

Dealing with many modalities

- Some variables can take many different values
 - CurrentURL
 - List of articles read
 - List of items seen

Idea 1: one-hot encoding + dictionary

- Associate each entry with an index i

- $\mathbf{x} = [0 \ 0 \ 0 \ \dots \ 0 \ \mathbf{1} \ 0 \ \dots \ 0 \ 0]$
 $\quad 0 \quad 1 \quad 2 \quad \quad \mathbf{i} \quad \quad (P-2) \ (P-1)$

Idea 1: one-hot encoding + dictionary

- Associate each entry with an index i

- $\mathbf{x} = [0 \ 0 \ 0 \ \dots \ 0 \ \mathbf{1} \ 0 \ \dots \ 0 \ 0]$
 0 1 2 *i* (P-2) (P-1)

- $\text{pCTR} = \sigma(\mathbf{w}^T \mathbf{x}) = \sigma(w_i)$

Building a dictionary

i	URL	w_i
0	http://google.com	-1.2
1	http://facebook.com	-3.4
...		
...		
129547171991	http://thiswebsiteisgreat.com	-0.5

Building a dictionary

i	URL	w_i
0	http://google.com	-1.2
1	http://facebook.com	-3.4
...		
...		
129547171991	http://thiswebsiteisgreat.com	-0.5
129547171992	http://thisoneisevenbetter.com	-0.45

Idea 2: using a hash table

i	w_i
0	-1.7
1	-2.1
...	
...	
...	
16777215	-1.2

- $h: S \rightarrow [0, 2^k - 1]$
- $h(\text{"http://google.com"})=14563$

Idea 2: using a hash table

i	w_i
0	-1.7
1	-2.1
...	
14563	-1.23
...	
16777215	-1.2

- $h: S \rightarrow [0, 2^k - 1]$
- $h(\text{"http://google.com"})=14563$

Collisions

- What if $h(S_0) = h(S_1)$?
- We will use the same w_i for both.
- This is called a **collision**.

Collisions in practice

- $h(\text{"http://google.com"}) = h(\text{"http://nicolas.le-roux.name"}) = 14563$
- $pCTR(\text{"http://google.com"}) = pCTR(\text{"http://nicolas.le-roux.name"})$
 $\approx CTR(\text{"http://google.com"})$

Example of a hash

- Current URL = `http://gouvernie.com/`
- $h(\text{"http://gouvernie.com/"}) = 12$
- $x = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]$
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Example of a hash

- Current URL = `http://gouvernie.com/` and Advertiser = S&W
- $h(\text{"http://gouvernie.com/"}) = 12$, $h(\text{" S&W "}) = 4$
- $x = [0\ 0\ 0\ 0\ \mathbf{1}\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ \mathbf{1}\ 0\ 0\ 0]$
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Limitations of the linear model

- $x = [0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]$

- $\text{pCTR} = \sigma(w^T x) = \frac{1}{1 + e^{-w^T x}} \approx e^{w^T x} = \prod_i e^{w_i x_i}$

Introducing cross-features

- Current URL = `http://gobernie.com/` and Advertiser = `S&W`
- $h(\text{"http://gobernie.com/" and " S&W "}) = 6$
- $x^{cf} = [0\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]$
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Cross-features as a second-order method

- $x = [0\ 0\ 0\ 0\ \mathbf{1}\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ \mathbf{1}\ 0\ 0\ 0]$

- $x^{cf} = [0\ 0\ 0\ 0\ \mathbf{1}\ 0\ \mathbf{1}\ 0\ 0\ 0\ 0\ 0\ 0\ \mathbf{1}\ 0\ 0\ 0]$

Cross-features as a second-order method

- $x = [0\ 0\ 0\ 0\ \mathbf{1}\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ \mathbf{1}\ 0\ 0\ 0]$

- $x^{cf} = [0\ 0\ 0\ 0\ \mathbf{1}\ 0\ \mathbf{1}\ 0\ 0\ 0\ 0\ 0\ 0\ \mathbf{1}\ 0\ 0\ 0]$

- $w^T x_{cf} = \sum_i w_i x_i$

Cross-features as a second-order method

- $x = [0\ 0\ 0\ 0\ \mathbf{1}\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ \mathbf{1}\ 0\ 0\ 0]$

- $x^{cf} = [0\ 0\ 0\ 0\ \mathbf{1}\ 0\ \mathbf{1}\ 0\ 0\ 0\ 0\ 0\ 0\ \mathbf{1}\ 0\ 0\ 0]$

- $w^T x_{cf} = \sum_i w_i x_i + \sum_{i,j} w_{ij} x_i x_j$

Cross-features as a second-order method

- $w^T x_{cf} = \sum_i w_i x_i + \sum_{i,j} w_{ij} x_i x_j$

- $w^T x_{cf} = w^T x + x^T M x$

The values in M are the same as those in w!

A matrix view of cross-features

- $pCTR = \sigma(x^T M x)$

M=

2.3	1.1	3.7	-3.0	1.1	2.3
-1.4	2.3	-3.0	3.7	-1.4	3.7
-3.0	-3.0	5.9	1.1	2.3	5.9
3.7	5.9	-1.4	1.1	-3.0	-1.4
-1.4	2.3	-1.4	-1.4	3.7	5.9
-3.0	1.1	1.1	5.9	5.9	5.9

The structure is determined by the hashing function

Exploiting the magic

"Thanks to hashing, the number of parameters in the model is **independent** of the number of variables. This means we should add as many variables as possible."

Reasons to NOT do that

- Because of collisions, adding variables may decrease performance
- Any variable needs to be **computed** and **stored**

The cost of adding variables

- *« Hey, I thought of this great variable: Time since last product view. Can we add it to the model? »*
- Storage: $\#Banners/day \times \#Days \times 4 = 480GB$
- RAM: $\#Users \times \#Campaigns \times 4 = 40GB$

Feature selection

- How to keep features while maintaining good performance? A tool to increase statistical efficiency
- Solution: selection of the optimal features and cross-features

Using sparsity-inducing regularizers

- $\min_w \sum_i l(w, x_i, y_i)$

Using sparsity-inducing regularizers

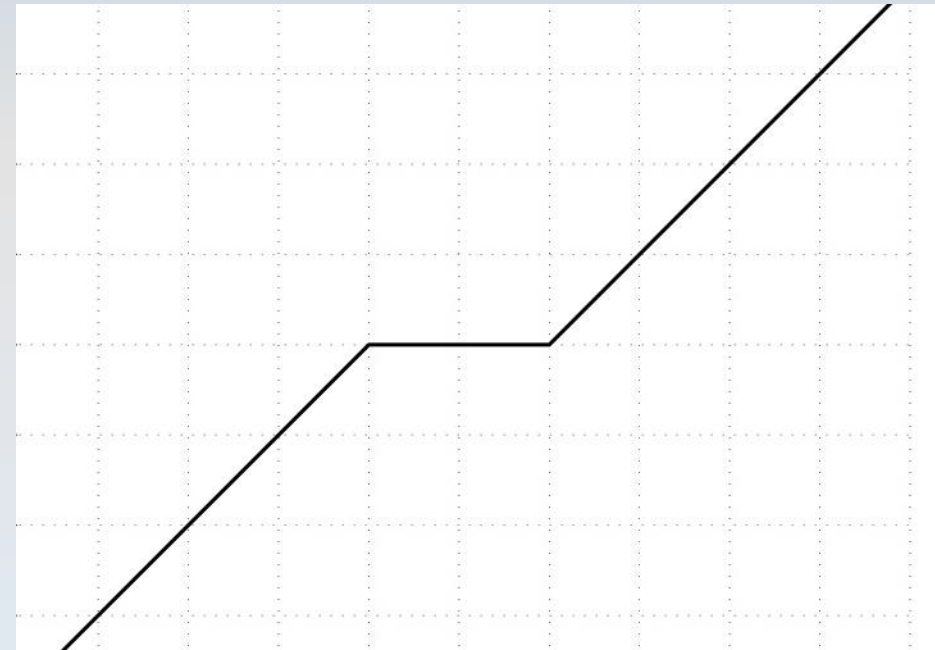
- $\min_w \sum_i l(w, x_i, y_i) + \lambda \|w\|_1$
- Statistically efficient
- Still requires to extract all variables

Using group-sparsity regularizers

- $\min_w \sum_i l(w, x_i, y_i) + \lambda \sum_g \|w_g\|_2$
- Forces all elements in a group to be 0
- The optimization problem remains efficient

Reducing bias

- Sparsity-inducing regularization introduces bias
- Two-stage process:
 - Select subset of variables
 - Re-optimize with the selected subset



Feature selection as kernel selection

- $w^T x_{cf} = w^T x + x^T M x$
- Doing feature selection on M is equivalent to learning the kernel

ML improves human efficiency

- Adding features is a critical part of an R&D
- Doing it automatically and well spares valuable people's time

Factorization machines

- $pCTR = \sigma(x^T M x)$

M=

2.3	1.1	2.3	-1.4	-3.0	3.7	-1.4	-3.0
-1.4	2.3	1.1	2.3	-3.0	5.9	2.3	1.1
-3.0	-3.0						
3.7	5.9						
-1.4	2.3						
-3.0	1.1						

Factorization machines

- $\phi(w, x) = w^T x$
- $\phi(M, x) = x^T M x$
- $\phi(U, x) = x^T U U^T x$

Linear model

	gobernie.com	drumpf4ever.com	hillaryous.com
S&W	$f(w_{bernie} + w_{S\&W})$	$f(w_{drumpf} + w_{S\&W})$	$f(w_{hillary} + w_{S\&W})$
Carebear	$f(w_{bernie} + w_{carebear})$	$f(w_{drumpf} + w_{carebear})$	$f(w_{hillary} + w_{carebear})$
JP Morgan	$f(w_{bernie} + w_{JPMorgan})$	$f(w_{drumpf} + w_{JPMorgan})$	$f(w_{hillary} + w_{JPMorgan})$

Level 2 cross-features

	gobernie.com	drumpf4ever.com	hillaryous.com
S&W	$f(w_{bernie,S\&W})$	$f(w_{drumpf,S\&W})$	$f(w_{hillary,S\&W})$
Carebear	$f(w_{bernie,carebear})$	$f(w_{drumpf,carebear})$	$f(w_{hillary,carebear})$
JP Morgan	$f(w_{bernie,JPMorgan})$	$f(w_{drumpf,JPMorgan})$	$f(w_{hillary,JPMorgan})$

Factorization machines

	gobernie.com	drumpf4ever.com	hillaryous.com
S&W	$f(\mathbf{w}_{bernie} \cdot \mathbf{w}_{S\&W})$	$f(\mathbf{w}_{drumpf} \cdot \mathbf{w}_{S\&W})$	$f(\mathbf{w}_{hillary} \cdot \mathbf{w}_{S\&W})$
Carebear	$f(\mathbf{w}_{bernie} \cdot \mathbf{w}_{carebear})$	$f(\mathbf{w}_{drumpf} \cdot \mathbf{w}_{carebear})$	$f(\mathbf{w}_{hillary} \cdot \mathbf{w}_{carebear})$
JP Morgan	$f(\mathbf{w}_{bernie} \cdot \mathbf{w}_{JPMorgan})$	$f(\mathbf{w}_{drumpf} \cdot \mathbf{w}_{JPMorgan})$	$f(\mathbf{w}_{hillary} \cdot \mathbf{w}_{JPMorgan})$

A side-by-side comparison

Standard cross-features

2.3	1.1	3.7	-3.0	1.1	2.3
-1.4	2.3	-3.0	3.7	-1.4	3.7
-3.0	-3.0	5.9	1.1	2.3	5.9
3.7	5.9	-1.4	1.1	-3.0	-1.4
-1.4	2.3	-1.4	-1.4	3.7	5.9
-3.0	1.1	1.1	5.9	5.9	5.9

- Frequent values are unregularized
- Infrequent modalities have random weights

Standard cross-features

2.3	1.1	2.3	-1.4	-3.0	3.7	-1.4	-3.0
-1.4	2.3	1.1	2.3	-3.0	5.9	2.3	1.1
-3.0	-3.0						
3.7	5.9						
-1.4	2.3						
-3.0	1.1						

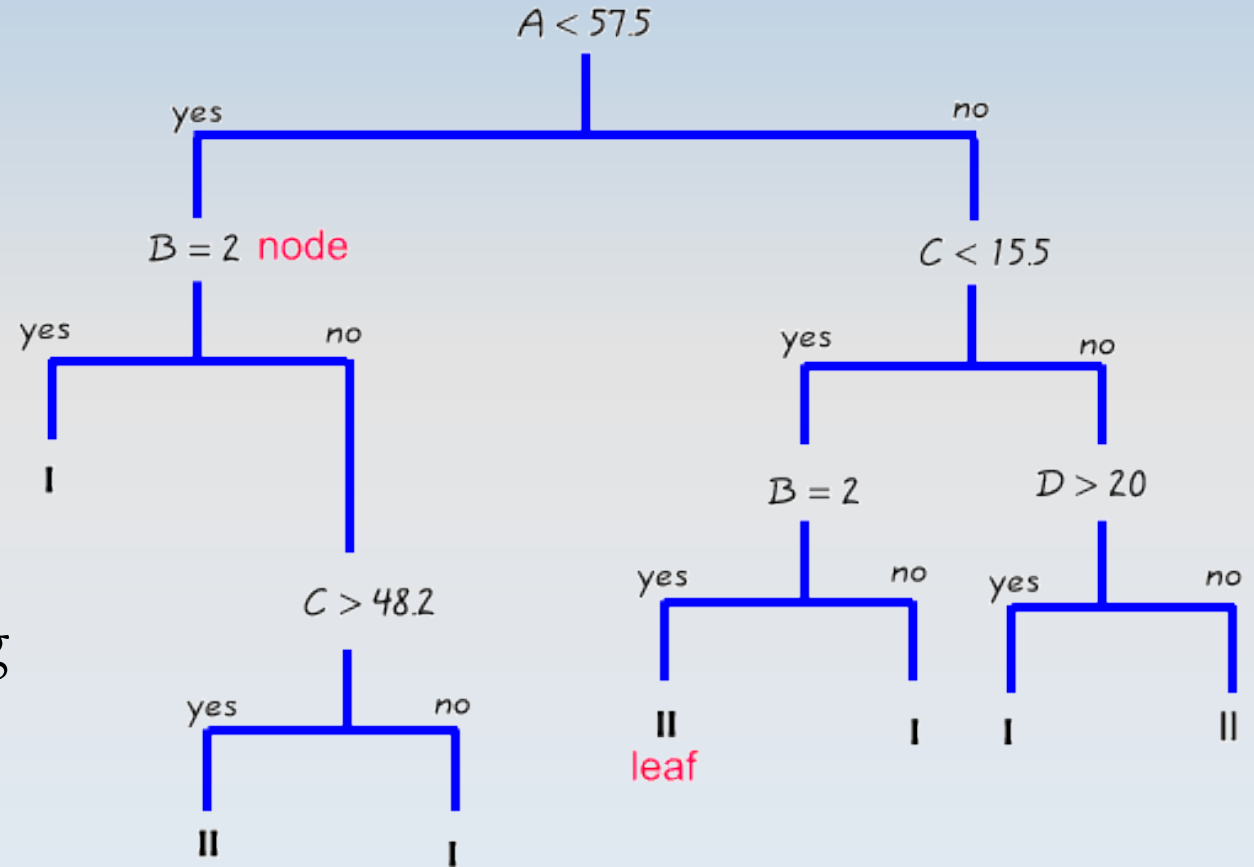
- All values are regularized

Handling continuous features

- Using a continuous feature directly only allows for linear interactions
- Finding the optimal transformation can be cumbersome

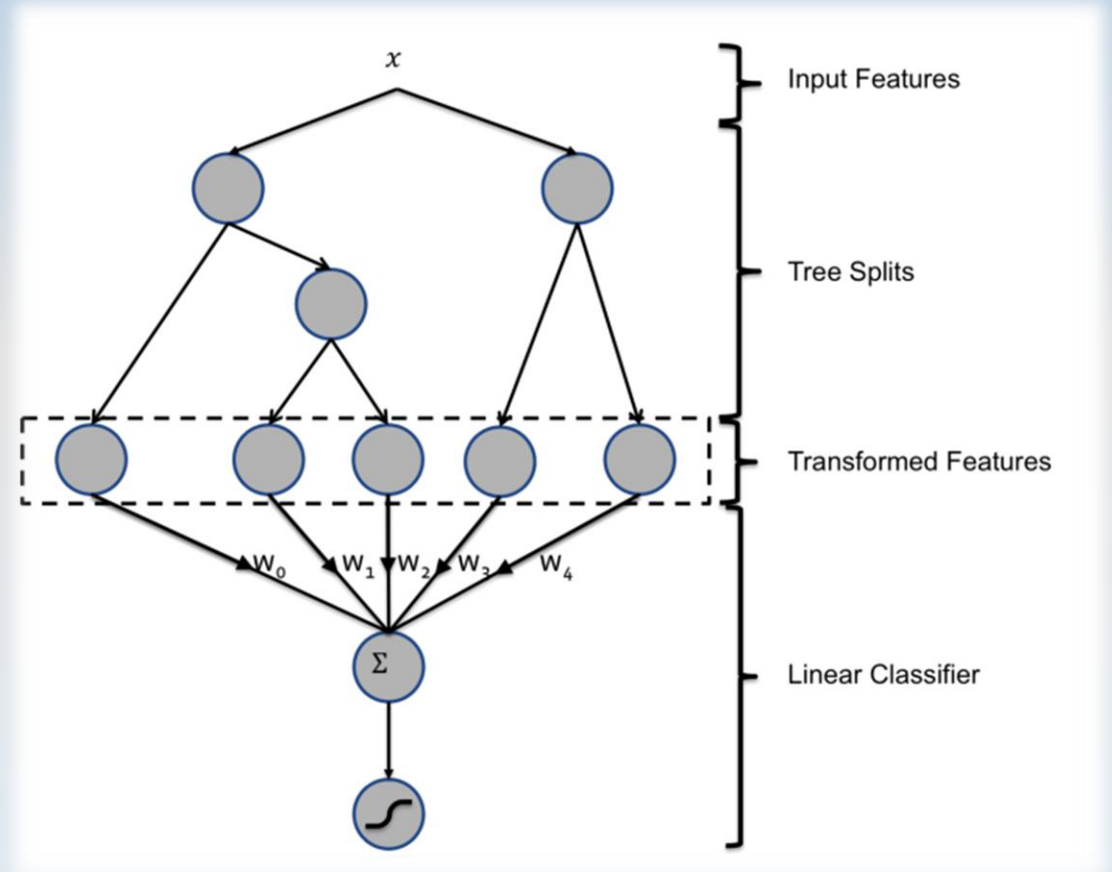
Gradient boosted decision trees

- Learn a decision tree to predict the clicks
- Learn a forest using boosting



Incorporating GBDT into a linear classifier

- Use the index of the leaves as categorical features



Learning the parameters

- $n = 10^9, p = 10^8$
- Theory tells us that stochastic gradient methods should be used

Arising optimization questions

- How do you set the stepsize for each of the 40 models?
- Does it change when we add features?
- How do you distribute the optimizer?
- Do all the datapoints have equal value?

Comparing the costs

- ML researcher: above 100k€ / year
- 16 CPUs - 64GB RAM: 5k€
- Win a factor 2 in 2 weeks

Further complications

- Increasing learning speed reduces delay
- But we still need to wait for the data
- And also for the log generation
- Learning time on a single machine at Criteo: 24 hours

A view of the entire pipeline

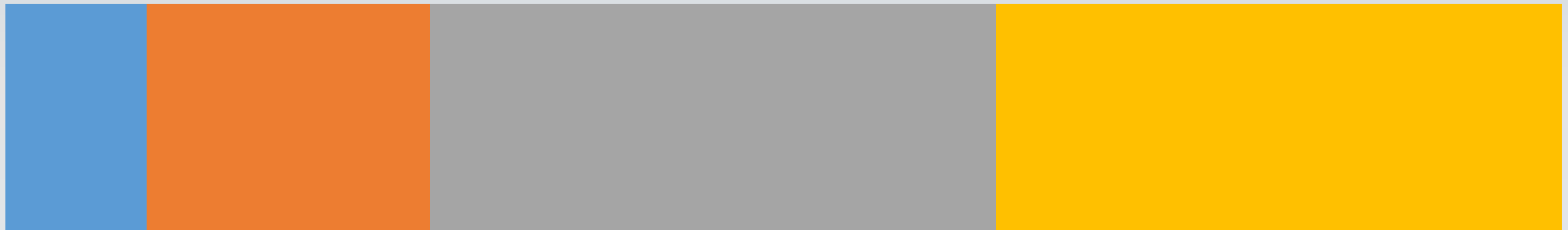


■ Gathering data

■ Generating logs

■ Learning the model

A view of the entire pipeline



■ Gathering data ■ Generating logs ■ Learning the model ■ Gain

A view of the entire pipeline



■ Gathering data ■ Generating logs ■ Learning the model ■ Gain

Focusing on the right problem

- After a bit, the return is too small
- It is important to identify when and to focus on other aspects
- Remember that what matters is the whole system

Comparison of optimization methods

Stochastic methods

- $O(1/T)$ convergence rate
- Cost independent of N
- "Faster" early on
- $O(1/T)$ on the test error

Batch methods

- $O(\rho^T)$ convergence rate
- Cost linear in N
- "Faster" later on
- $O(1/T)$ on the test error

Real comparison of optimization methods

Stochastic methods

- Careful with the stepsize!
- Hire a team to distribute it
- "Faster" early on

Batch methods

- Line-search and forget
- 10 lines of code to distribute
- Initialize properly

Robustness trumps accuracy

Criteo's optimizer

- Distributed L-BFGS
- Distributed computation of the gradients (10^7 examples/s)
- Update computation on a single node

Automatic hyperparameter optimization

- Number of hyperparameters grows w/ complexity of the model
- Optimizing them efficiently can have a huge impact
- Current approaches use GPs to model the test error as a function of their values

Noisy targets

- So far, we focused on a click prediction model
- It is probably not what we want
- The true goal is the (incremental) sale



Predicting sales

- There are far fewer sales than clicks (1 sale for 10 000 displays)
- They come after 30 days

Approximating 30-day sales

- We can use sales over a shorter period
- This leads to biased prediction
- What else can we do?

Modeling delayed feedback

- E = elapsed time since the click
- D = delay between the click and the sale
- Y = did the sale already occur?
- C = will a sale eventually occur?
- Build a joint model $P(C, D)$

Modeling delayed feedback

- $P(C)$: probability that a sale will occur
- $P(D | C=1)$: probability of observing a delay D for occurring sales
- If $Y=0$ after elapsed time E , then

$$P(C=1 | Y=0, E) = \int_{D>E} P(C = 1, D)dD$$

From unsupervised to weakly supervised learning

- Unsupervised learning tries to learn about the input data
- Weakly supervised learning uses related tasks
 - Long visits on the website
 - Sales which do not follow a click
- Big data: unstructured targets rather than inputs