Polynomial Networks and Factorization Machines: New Insights and Efficient Training Algorithms

Mathieu Blondel

NTT Communication Science Laboratories
Kyoto, Japan

Joint work with M. Ishihata, A. Fujino and N. Ueda

Presented at ICML 2016

2016/8/10

Supervised learning with polynomials

- From $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$, $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, learn a polynomial

$$\hat{y} : \mathbb{R}^d \to \mathbb{R}$$

- Motivation
  - **Universality**: polynomials can approximate any $\hat{y} : \mathbb{R}^d \to \mathbb{R}$ arbitrary well on a compact subset of $\mathbb{R}^d$ (Stone-Weierstrass theorem)
  - **Interpretability**: Feature combinations are meaningful in many applications (NLP, bioinformatics, etc)

Polynomial regression

- Assign weights to feature combinations

$$\hat{y}_{\mathrm{PR}}(\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{W}) := \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \sum_{j' > j} \boldsymbol{W}_{j,j'} x_j x_{j'}$$

  where $\boldsymbol{w} \in \mathbb{R}^d$ and $\boldsymbol{W} \in \mathbb{R}^{d \times d}$

- Pro: reduces to a simple linear model
- Con: does not scale well to high-dimensional data

Kernel methods for polynomial regression

- Use a **polynomial kernel** so as to **implicitly map** the data to feature combinations via the kernel trick
- Predictions are computed by

Linear dependence on training set size!

$$\hat{y}_{\mathrm{KM}}(\boldsymbol{x}; \boldsymbol{\alpha}) := \sum_{i=1}^{n} \alpha_i \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x})$$

where $\boldsymbol{\alpha} \in \mathbb{R}^n$ and $\mathcal{K}$ is set to

$$\mathcal{P}_{\gamma}^{m}(\boldsymbol{x}_i, \boldsymbol{x}) := (\gamma + \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle)^m$$

Factorization machines (Rendle 2010)

- Recall polynomial regression

$$\hat{y}_{\mathrm{PR}}(\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{W}) := \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \sum_{j' > j} \boldsymbol{W}_{j,j'} x_j x_{j'}$$

- In FMs, we replace $\boldsymbol{W} \in \mathbb{R}^{d \times d}$ by a **factorized** matrix

$$\hat{y}_{\mathrm{FM}}(\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{P}) := \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \sum_{j' > j} (\boldsymbol{P}\boldsymbol{P}^{\mathrm{T}})_{j,j'} x_j x_{j'}$$

$$\boldsymbol{w} \in \mathbb{R}^{d}, \quad \boldsymbol{P} \in \mathbb{R}^{d \times k} \quad k \ll d$$

FMs: pros and cons

- ☺ Reduced number of parameters to estimate

  $O(dk)$ instead of $O(d^2)$ (PR) or $O(n)$ (KM)

- ☺ Faster predictions

  $O(dk)$ instead of $O(d^2)$ (PR) or $O(dn)$ (KM)

- ☺ Ability to infer weight of unobserved feature combinations (useful for recommender systems)

- ☹ Learning $\boldsymbol{P}$ involves a non-convex problem

Proposed framework

- We consider models of the form

$$\hat{y}_{\mathcal{K}}(\boldsymbol{x}; \boldsymbol{\lambda}, \boldsymbol{P}) := \sum_{s=1}^{k} \lambda_s \mathcal{K}(\boldsymbol{p}_s, \boldsymbol{x})$$

  where $\boldsymbol{\lambda} \in \mathbb{R}^k$ and $\boldsymbol{P} \in \mathbb{R}^{d \times k}$ with columns $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_k$

- We focus on two kernels:
  - ANOVA kernel (recover factorization machines)
  - Homogeneous polynomial kernel (recover "polynomial networks")

Polynomial and ANOVA kernels ($m = 2$)

- Homogeneous polynomial kernel

$$\mathcal{H}^2(\boldsymbol{p}, \boldsymbol{x}) := \langle \boldsymbol{p}, \boldsymbol{x} \rangle^2 = \sum_{i,j=1}^{d} p_i x_i p_j x_j$$

Uses **all** feature combinations: x_i^2 and $x_i x_j$ for $i \neq j$

- ANOVA kernel (Vapnik 1998)

$$\mathcal{A}^2(\boldsymbol{p}, \boldsymbol{x}) := \sum_{j>i} p_i x_i p_j x_j$$

Uses **distinct** feature combinations: $x_i x_j$ for $i \neq j$

Polynomial and ANOVA kernels ($m = 3$)

- Homogeneous polynomial kernel

$$\mathcal{H}^3(\boldsymbol{p}, \boldsymbol{x}) := \langle \boldsymbol{p}, \boldsymbol{x} \rangle^3 = \sum_{i,j,k=1}^{d} p_i x_i p_j x_j p_k x_k$$

Uses **all** feature combinations: x_i^3, $x_i^2 x_j$, and $x_i x_j x_k$

- ANOVA kernel (Vapnik 1998)

$$\mathcal{A}^3(\boldsymbol{p}, \boldsymbol{x}) := \sum_{k>j>i} p_i x_i p_j x_j p_k x_k$$

Uses **distinct** feature combinations: $x_i x_j x_k$ for $i \neq j \neq k$

Polynomial and ANOVA kernels ($m \geq 2$)

- Homogeneous polynomial kernel

$$\mathcal{H}^m(\boldsymbol{p}, \boldsymbol{x}) := \langle \boldsymbol{p}, \boldsymbol{x} \rangle^m = \sum_{j_1, \ldots, j_m = 1}^{d} p_{j_1} x_{j_1} \ldots p_{j_m} x_{j_m}$$

Uses **all** feature combinations (**with** replacement)

- ANOVA kernel (Vapnik 1998)

$$\mathcal{A}^m(\boldsymbol{p}, \boldsymbol{x}) := \sum_{j_m > \cdots > j_1} p_{j_1} x_{j_1} \ldots p_{j_m} x_{j_m}$$

Uses **distinct** feature combinations (**without** replacement)

Expressing FMs and PNs using kernels

- Recall that

$$\hat{y}_{\mathcal{K}}(\boldsymbol{x}; \boldsymbol{\lambda}, \boldsymbol{P}) := \sum_{s=1}^{k} \lambda_s \mathcal{K}(\boldsymbol{p}_s, \boldsymbol{x})$$

- Expressing factorization machines

$$\hat{y}_{\text{FM}}(\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{P}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \hat{y}_{\mathcal{A}^2}(\boldsymbol{x}; \mathbf{1}, \boldsymbol{P})$$

- Expressing polynomial networks

$$\hat{y}_{\text{PN}}(\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{P}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \hat{y}_{\mathcal{H}^2}(\boldsymbol{x}; \boldsymbol{\lambda}, \boldsymbol{P})$$

Direct optimization

- Most natural approach: directly minimize

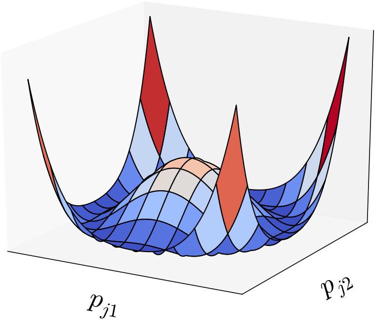
$$D_{\mathcal{K}}(\boldsymbol{\lambda}, \boldsymbol{P}) := \sum_{i=1}^{n} \ell \left(y_i, \sum_{s=1}^{k} \lambda_s \mathcal{K}(\boldsymbol{p}_s, \boldsymbol{x}_i) \right) + \beta |\lambda_s| \, \|\boldsymbol{p}_s\|^2$$

where ℓ is a μ-smooth convex loss function

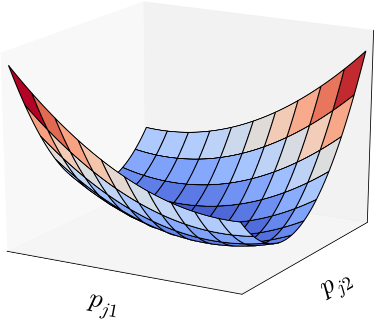
- **Convexity?**

	$\mathcal{K} = \mathcal{H}^m$	$\mathcal{K} = \mathcal{A}^m$	
$\boldsymbol{\lambda}$	convex	convex	
$\boldsymbol{P}$	non-convex	non-convex	
rows of $\boldsymbol{P}$	non-convex	convex	← thanks to multi-linearity of $\mathcal{A}^m$
columns of $\boldsymbol{P}$	non-convex	non-convex	
elements of $\boldsymbol{P}$	non-convex	convex	

Direct optimization



(a) when $\mathcal{K} = \mathcal{H}^2$



(b) when $\mathcal{K} = \mathcal{A}^2$

Objective function w.r.t. one row of $\boldsymbol{P}$

Multi-convex optimization

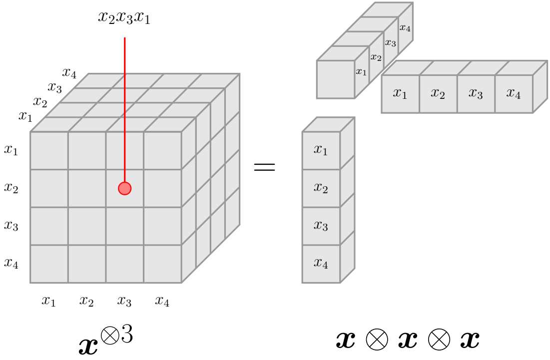
- When $\mathcal{K} = \mathcal{A}^m$, the objective is called **multi-convex**
- We can use alternating minimization
  - Popular in the matrix and tensor factorization literature
  - Simple to implement
  - Converges to a stationary point
  - When ℓ is the squared loss, each sub-problem can be solved analytically

A tensor approach

- When $\mathcal{K} = \mathcal{H}^m$, the direct objective is **neither** convex **nor** multi-convex
- We will now present an objective that is multi-convex for **both** $\mathcal{K} = \mathcal{H}^m$ and $\mathcal{A}^m$
- The main idea is to convert the estimation of $\boldsymbol{\lambda}$ and $\boldsymbol{P}$ to that of a **low-rank symmetric tensor** $\boldsymbol{\mathcal{W}}$

Rank-one symmetric tensor

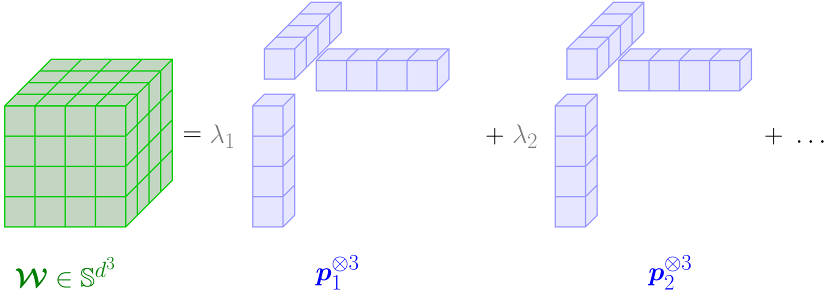
$$\boldsymbol{x}^{\otimes m} := \underbrace{\boldsymbol{x} \otimes \cdots \otimes \boldsymbol{x}}_{m \text{ times}} \in \mathbb{S}^{d^m}$$



Symmetric tensor decomposition

$$\mathcal{W} = \sum_{s=1}^{k} \lambda_s \boldsymbol{p}_s^{\otimes m}$$

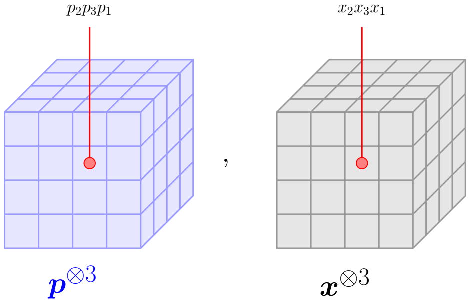
where k is the (symmetric) rank of $\mathcal{W}$



Link between tensors and poly. kernel

Homogeneous polynomial kernel can be rewritten as

$$\mathcal{H}^m(\boldsymbol{p}, \boldsymbol{x}) := \langle \boldsymbol{p}, \boldsymbol{x} \rangle^m = \langle \boldsymbol{p}^{\otimes m}, \boldsymbol{x}^{\otimes m} \rangle$$

$\mathcal{H}^3(\boldsymbol{p}, \boldsymbol{x}) = \langle$  $\rangle$

Link between tensors and ANOVA kernel

- For the ANOVA kernel, we need to **ignore irrelevant feature combinations**...
- We introduce the following notation

$$\langle \mathcal{W}, \mathcal{X} \rangle_{>} := \sum_{j_m > \cdots > j_1} \mathcal{W}_{j_1,\ldots,j_m} \mathcal{X}_{j_1,\ldots,j_m} \qquad \mathcal{W}, \mathcal{X} \in \mathbb{S}^{d^m}$$

- Then

$$\mathcal{A}^m(\boldsymbol{p}, \boldsymbol{x}) = \langle \boldsymbol{p}^{\otimes m}, \boldsymbol{x}^{\otimes m} \rangle_{>}$$

Link between tensors and kernel expansions

↓ not multi-linear ☹

- Assume $\mathcal{W}$ is decomposed as $\sum_{s=1}^{k} \lambda_s \boldsymbol{p}_s^{\otimes m}$. Then,

$$\hat{y}_{\mathcal{H}^2} = \langle \mathcal{W}, \boldsymbol{x}^{\otimes m} \rangle = \sum_{s=1}^{k} \lambda_s \mathcal{H}^m(\boldsymbol{p}_s, \boldsymbol{x})$$

$$\hat{y}_{\mathcal{A}^2} = \langle \mathcal{W}, \boldsymbol{x}^{\otimes m} \rangle_{>} = \sum_{s=1}^{k} \lambda_s \mathcal{A}^m(\boldsymbol{p}_s, \boldsymbol{x})$$

- We can convert the estimation of $\boldsymbol{\lambda}$ and $\boldsymbol{P}$ to that of a low-rank tensor $\mathcal{W}$

Key idea of the proposed method

- Expressing the loss as a function of $\mathcal{W}$

$$L_{\mathcal{H}^m}(\mathcal{W}) := \sum_{i=1}^{n} \ell\left(y_i, \langle \mathcal{W}, \boldsymbol{x}_i^{\otimes m} \rangle\right)$$

$$L_{\mathcal{A}^m}(\mathcal{W}) := \sum_{i=1}^{n} \ell\left(y_i, \langle \mathcal{W}, \boldsymbol{x}_i^{\otimes m} \rangle_{>}\right)$$

- Our idea: we set $\mathcal{W} = \mathcal{S}\left(\sum_{s=1}^{r} \boldsymbol{u}_s^1 \otimes \cdots \otimes \boldsymbol{u}_s^m\right)$ ($\downarrow$ multi-linear ☺)

  where $\mathcal{S}(\mathcal{M})$ is the symmetrization of $\mathcal{M}$

Multi-convex formulation

$$\min_{\boldsymbol{U}^1,\ldots,\boldsymbol{U}^m \in \mathbb{R}^{d\times r}} L_{\mathcal{K}}\left(\mathcal{S}\left(\sum_{s=1}^{r} \boldsymbol{u}_s^1 \otimes \cdots \otimes \boldsymbol{u}_s^m\right)\right) + \frac{\beta}{2}\sum_{t=1}^{m} \|\boldsymbol{U}^t\|_F^2$$

where $\boldsymbol{u}_s^t$ is s^{th} column of $\boldsymbol{U}^t$

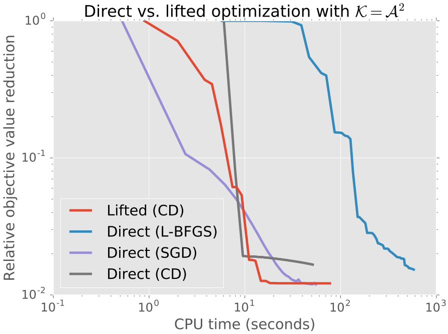
- Convex in $\boldsymbol{U}^1, \ldots, \boldsymbol{U}^m$ separately due to **multi-linearity**
- When $m = 2$, this is equivalent to direct formulation (and we can easily convert $\boldsymbol{U}^1$, $\boldsymbol{U}^2$ to λ, $\boldsymbol{P}$)
- Coordinate descent: costs $O(mrn_z(\boldsymbol{X}))$ per epoch

Direct vs. proposed approach

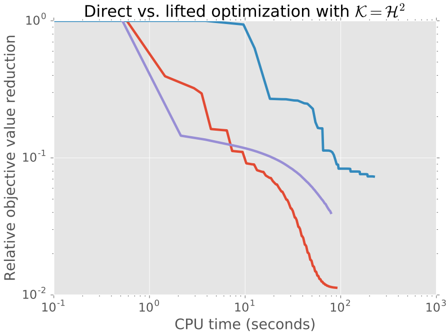
	Direct	Proposed
Parameters	$\boldsymbol{\lambda} \in \mathbb{R}^k$ $\boldsymbol{P} \in \mathbb{R}^{d \times k}$	$\boldsymbol{U}^1, \dots, \boldsymbol{U}^m \in \mathbb{R}^{d \times r}$
Multi-convex if	$\mathcal{K} = \mathcal{A}^m$	$\mathcal{K} = \mathcal{A}^m$ or $\mathcal{H}^m$
Multi-convex in	$\boldsymbol{\lambda}$ and rows of $\boldsymbol{P}$	$\boldsymbol{U}^1, \dots, \boldsymbol{U}^m$

In practice, we set $r = k/m$.

Direct vs. proposed (“lifted”)



(a) $\mathcal{K} = \mathcal{A}^2$



(b) $\mathcal{K} = \mathcal{H}^2$

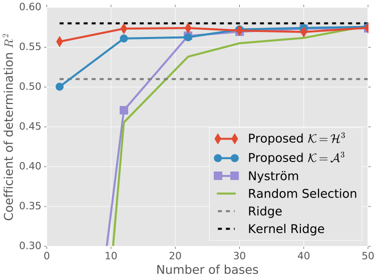
E2006-tfidf dataset

$n = 16,087,\ d = 150,360$

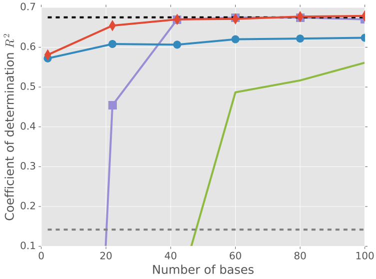
Low-budget non-linear regression

We compared six methods:

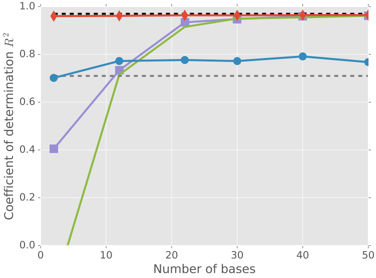
1. Proposed with $\mathcal{K} = \mathcal{H}^3$ (with $\boldsymbol{x}^{\mathrm{T}} \leftarrow [1, \boldsymbol{x}^{\mathrm{T}}]$),
2. Proposed with $\mathcal{K} = \mathcal{A}^3$ (with $\boldsymbol{x}^{\mathrm{T}} \leftarrow [1, \boldsymbol{x}^{\mathrm{T}}]$),
3. Nyström method with $\mathcal{K} = \mathcal{P}_\gamma^3$, where $\gamma = 1$
4. Random Selection: choose bases uniformly at random from training set with $\mathcal{K} = \mathcal{P}_\gamma^3$.
5. Linear ridge regression
6. Kernel ridge regression with $\mathcal{K} = \mathcal{P}_\gamma^3$



(a) abalone



(b) cadata



(c) cpusmall

Conclusion

- We proposed a **unified** framework for factorization machines (FM) and polynomial networks (PN)
- We proposed efficient training algorithms based on **tensor decomposition**

Open-source implementation by Vlad Niculae:
`http://contrib.scikit-learn.org/polylearn/`