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

Mathieu Blondel MATHIEU.BLONDEL@LAB.NTT.CO.JP
Masakazu Ishihata ISHIHATA.MASAKAZU@LAB.NTT.CO.JP
Akinori Fujino FUINO.AKINORI@LAB.NTT.CO.JP
Naonori Ueda UEDA.NAONORI@LAB.NTT.CO.JP

NTT Communication Science Laboratories, 2-4 Hikaridai Seika-cho Soraku-gun, Kyoto 619-0237, Japan

Abstract

Polynomial networks and factorization machines are two recently-proposed models that can efficiently use feature interactions in classification and regression tasks. In this paper, we revisit both models from a unified perspective. Based on this new view, we study the properties of both models and propose new efficient training algorithms. Key to our approach is to cast parameter learning as a low-rank symmetric tensor estimation problem, which we solve by multi-convex optimization. We demonstrate our approach on regression and recommender system tasks.

1. Introduction

Interactions between features play an important role in many classification and regression tasks. One of the simplest approaches to leverage such interactions consists in explicitly augmenting feature vectors with products of features (monomials), as in polynomial regression. Although fast linear model solvers can be used (Chang et al., 2010; Sonnenburg & Franc, 2010), an obvious drawback of this kind of approach is that the number of parameters to estimate scales as $O(d^m)$, where $d$ is the number of features and $m$ is the order of interactions considered. As a result, it is usually limited to second or third-order interactions.

Another popular approach consists in using a polynomial kernel so as to implicitly map the data via the kernel trick. The main advantage of this approach is that the number of parameters to estimate in the model is actually independent of $d$ and $m$. However, the cost of storing and evaluating the model is now proportional to the number of training instances. This is sometimes called the curse of kernelization (Wang et al., 2010). Common ways to address the issue include the Nyström method (Williams & Seeger, 2001), random features (Kar & Karnick, 2012) and sketching (Pham & Pagh, 2013; Avron et al., 2014).

In this paper, in order to leverage feature interactions in possibly very high-dimensional data, we consider models which predict the output $y \in \mathbb{R}$ associated with an input vector $x \in \mathbb{R}^d$ by

$$\hat{y}_K(x; \lambda, P) := \sum_{s=1}^{k} \lambda_s K(p_s, x),$$

(1)

where $\lambda = [\lambda_1, \ldots, \lambda_k]^T \in \mathbb{R}^k$, $P = [p_1, \ldots, p_k] \in \mathbb{R}^{d \times k}$, $K$ is a kernel and $k$ is a hyper-parameter. More specifically, we focus on two specific choices of $K$ which allow us to use feature interactions: the homogeneous polynomial and the ANOVA kernels. Our contributions are as follows. We show (Section 3) that choosing one kernel or the other allows us to recover polynomial networks (PNs) (Livni et al., 2014) and, surprisingly, factorization machines (FMs) (Rendle, 2010; 2012). Based on this new view, we show important properties of PNs and FMs. Notably, we show for the first time that the objective function of arbitrary-order FMs is multi-convex (Section 4). Unfortunately, the objective function of PNs is not multi-convex. To remedy this problem, we propose a lifted approach, based on casting parameter estimation as a low-rank tensor estimation problem (Section 5.1). Combined with a symmetrization trick, this approach leads to a multi-convex problem, for both PNs and FMs (Section 5.2). We demonstrate our approach on regression and recommender system tasks.

Notation. We denote vectors, matrices and tensors using lower-case, upper-case and calligraphic bold, e.g., $w$, $W$ and $\mathcal{W}$. We denote the set of $\mathbb{R}^{d \times \cdots \times d}$ real tensors by $\mathbb{R}^{d^m}$ and the set of symmetric real tensors by $\mathbb{S}^{d^m}$. We use $\langle \cdot, \cdot \rangle$ to denote vector, matrix and tensor inner product. Given $x$, we define a symmetric rank-one tensor by
\(x^{\otimes m} := x \otimes \cdots \otimes x \in \mathbb{R}^{d^m}\), where \((x^{\otimes m})_{j_1,j_2, \ldots, j_m} = x_{j_1}x_{j_2} \cdots x_{j_m}\). We use \([d]\) to denote the set \(\{1, \ldots, d\}\).

2. Related work

2.1. Polynomial networks

Polynomial networks (PNs) (Livni et al., 2014) of degree \(m = 2\) predict the output \(y \in \mathbb{R}\) associated with \(x \in \mathbb{R}^d\) by

\[
\hat{y}_{\text{PN}}(x; w, \lambda, P) := \langle w, x \rangle + \sigma(P^T x),
\]

(2)

where \(w \in \mathbb{R}^d\), \(P \in \mathbb{R}^{d \times k}\), \(\lambda \in \mathbb{R}^k\) and \(\sigma(u) := u^2\) is evaluated element-wise. Intuitively, the right-hand term can be interpreted as a feedforward neural network with one hidden layer of \(k\) units and with activation function \(\sigma(u)\). Livni et al. (2014) also extend (2) to the case \(m = 3\) and show theoretically that PNs can approximate feedforward networks with sigmoidal activation. A similar model was independently shown to perform well on dependency parsing (Chen & Manning, 2014). Unfortunately, the objective function of PNs is non-convex. In Section 5, we derive a multi-convex objective based on low-rank symmetric tensor estimation, suitable for training arbitrary-order PNs.

2.2. Factorization machines

One of the simplest way to leverage feature interactions is polynomial regression (PR). For example, for second-order interactions, in this approach, we compute predictions by

\[
\hat{y}_{\text{PR}}(x; w, W) := \langle w, x \rangle + \sum_{j' > j} W_{j,j'}x_jx_{j'},
\]

where \(w \in \mathbb{R}^d\) and \(W \in \mathbb{R}^{d \times d}\). Obviously, model size in PR does not scale well w.r.t. \(d\). The main idea of (second-order) factorization machines (FMs) (Rendle, 2010; 2012) is to replace \(W\) with a factorized matrix \(PP^T\):

\[
\hat{y}_{\text{FM}}(x; w, P) := \langle w, x \rangle + \sum_{j' > j} (PP^T)_{j,j'}x_jx_{j'},
\]

where \(P \in \mathbb{R}^{d \times k}\). FMs have been increasingly popular for efficiently modeling feature interactions in high-dimensional data, see (Rendle, 2012) and references therein. In Section 4, we show for the first time that the objective function of arbitrary-order FMs is multi-convex.

3. Polynomial and ANOVA kernels

In this section, we show that the prediction functions used by polynomial networks and factorization machines can be written using (1) for a specific choice of kernel.

The polynomial kernel is a popular kernel for using combinations of features. The kernel is defined as

\[
P_m^\gamma(x, x) := (\gamma + \langle x, x \rangle)^m,
\]

where \(m \in \mathbb{N}\) is the degree and \(\gamma > 0\) is a hyper-parameter. We define the homogeneous polynomial kernel by

\[
\mathcal{H}^m(p, x) := \mathcal{P}_m^0(p, x) = \langle p, x \rangle^m.
\]

Let \(p = [p_1, \ldots, p_d]^T\) and \(x = [x_1, \ldots, x_d]^T\). Then,

\[
\mathcal{H}^m(p, x) = \sum_{j_1=1}^d \cdots \sum_{j_m=1}^d p_{j_1}x_{j_1} \cdots p_{j_m}x_{j_m}.
\]

We thus see that \(\mathcal{H}^m\) uses all monomials of degree \(m\) (i.e., all combinations of features with replacement).

A much lesser known kernel is the ANOVA kernel (Stitson et al., 1997; Vapnik, 1998). Following (Shawe-Taylor & Cristianini, 2004, Section 9.2), the ANOVA kernel of degree \(m\), where \(2 \leq m \leq d\), can be defined as

\[
\mathcal{A}^m(p, x) := \sum_{j_m > \cdots > j_1} p_{j_1}x_{j_1} \cdots p_{j_m}x_{j_m}.
\]

As a result, \(\mathcal{A}^m\) uses only monomials composed of distinct features (i.e., feature combinations without replacement). For later convenience, we also define \(\mathcal{A}^1(p, x) := 1\) and \(\mathcal{A}^1(p, x) := \langle p, x \rangle\).

With \(\mathcal{H}^m\) and \(\mathcal{A}^m\) defined, we are now in position to state the following lemma.

Lemma 1 Expressing PNs and FMs using kernels

Let \(\hat{y}_K(x; \lambda, P)\) be defined as in (1). Then,

\[
\hat{y}_{\text{PR}}(x; w, P) := \langle w, x \rangle + \sigma(P^T x),
\]

\[
\hat{y}_{\text{FM}}(x; w, P) := \langle w, x \rangle + \sum_{j' > j} (PP^T)_{j,j'}x_jx_{j'},
\]

The relation easily extends to higher orders. This new view allows us to state results that will be very useful in the next sections. The first one is that \(\mathcal{H}^m\) and \(\mathcal{A}^m\) are homogeneous functions, i.e., they satisfy

\[
\lambda^m \mathcal{K}(\lambda p, x) = \mathcal{K}(\lambda p, x) \quad \forall \lambda \in \mathbb{R}, \forall m \in \mathbb{N}_+.
\]

Another key property of \(\mathcal{A}^m(p, x)\) is multi-linearity.\(^1\)

Lemma 2 Multi-linearity of \(\mathcal{A}^m(p, x)\) w.r.t. \(p_1, \ldots, p_d\)

Let \(p, x \in \mathbb{R}^d\), \(j \in [d]\) and \(1 \leq m \leq d\). Then,

\[
\mathcal{A}^m(p, x) = \mathcal{A}^m(p_{-j}, x_{-j}) + p_jx_j \mathcal{A}^{m-1}(p_{-j}, x_{-j})
\]

where \(p_{-j}\) denotes the \((d-1)\)-dimensional vector with \(p_j\) removed and similarly for \(x_{-j}\).

That is, everything else kept fixed, \(\mathcal{A}^m(p, x)\) is an affine function of \(p_j\), \(\forall j \in [d]\). Proof is given in Appendix B.1.

\(^1\)A function \(f(\theta_1, \ldots, \theta_k)\) is called multi-linear (resp. multi-convex) if it is linear (resp. convex) w.r.t. \(\theta_1, \ldots, \theta_k\) separately.
Assuming \( p \) is dense and \( x \) sparse, the cost of naively computing \( \mathcal{A}^m(p, x) \) by (3) is \( O(n_x(x)^m) \), where \( n_x(x) \) is the number of non-zero features in \( x \). To address this issue, we will make use of the following lemma for computing \( \mathcal{A}^m \) in nearly \( O(mn_x(x)) \) time when \( m \in \{2,3\} \).

**Lemma 3** Effcient computation of ANOVA kernel

\[
\begin{align*}
A^2(p, x) &= \frac{1}{2} \left[ H^2(p, x) - D^2(p, x) \right] \\
A^3(p, x) &= \frac{1}{6} \left[ H^3(p, x) - 3D^{2,1}(p, x) + 2D^3(p, x) \right]
\end{align*}
\]

where we defined \( D^m(p, x) := \sum_{j=1}^d (p_jx_j)^m \) and \( D^{m,n}(p, x) := D^m(p, x)D^n(p, x) \).

See Appendix B.2 for a derivation.

**4. Direct approach**

Let us denote the training set by \( X = [x_1, \ldots, x_n] \in \mathbb{R}^{d \times n} \) and \( y = [y_1, \ldots, y_n]^T \in \mathbb{R}^n \). The most natural approach to learn models of the form (1) is to directly choose \( \lambda \) and \( P \) so as to minimize some error function

\[
D_K(\lambda, P) := \sum_{i=1}^n \ell(y_i, \hat{y}_K(x_i; \lambda, P)),
\]

where \( \ell(y_i, \hat{y}_i) \) is a convex loss function. Note that (4) is a convex objective w.r.t. \( \lambda \) regardless of \( K \). However, it is in general non-convex w.r.t. \( P \). Fortunately, when \( K = \mathcal{A}^m \), we can show that (4) is multi-convex.

**Theorem 1** Multi-convexity of (4) when \( K = \mathcal{A}^m \)

\( D_{\mathcal{A}^m} \) is convex in \( \lambda \) and in each row of \( P \) separately.

Proof is given in Appendix B.3. As a corollary, the objective function of FMs of arbitrary order is thus multi-convex.

Theorem 1 suggests that we can minimize (4) efficiently when \( K = \mathcal{A}^m \) by solving a succession of convex problems w.r.t. \( \lambda \) and the rows of \( P \). We next show that when \( m \) is odd, we can just fix \( \lambda = 1 \) without loss of generality.

**Lemma 4** When is it useful to fit \( \lambda \)?

Let \( \mathcal{K} = \mathcal{H}^m \) or \( \mathcal{A}^m \). Then

\[
\begin{align*}
\min_{\lambda \in \mathbb{R}^k, P \in \mathbb{R}^{d \times k}} D_K(\lambda, P) &\leq \min_{P \in \mathbb{R}^{d \times k}} D_K(1, P) \quad \text{if } m \text{ is even} \\
\min_{\lambda \in \mathbb{R}^k, P \in \mathbb{R}^{d \times k}} D_K(\lambda, P) &= \min_{P \in \mathbb{R}^{d \times k}} D_K(1, P) \quad \text{if } m \text{ is odd}.
\end{align*}
\]

The result stems from the fact that \( \mathcal{H}^m \) and \( \mathcal{A}^m \) are homogeneous functions. If we define \( v := \text{sign}(\lambda) \sqrt{\lambda} p \), then we obtain \( \lambda \mathcal{H}^m(p, x) = \mathcal{H}^m(v, x) \forall \lambda \) if \( m \) is odd, and similarly for \( \mathcal{A}^m \). That is, \( \lambda \) can be absorbed into \( v \) without loss of generality. When \( m \) is even, \( \lambda < 0 \) cannot be absorbed unless we allow complex numbers. Because FMs fix \( \lambda = 1 \), Lemma 4 shows that the class of functions that FMs can represent is possibly smaller than our framework.

**5. Lifted approach**

**5.1. Conversion to low-rank tensor estimation problem**

If we set \( \mathcal{K} = \mathcal{H}^m \) in (4), the resulting optimization problem is neither convex nor multi-convex w.r.t. \( P \). In (Blondel et al., 2015), for \( m = 2 \), it was proposed to cast parameter estimation as a low-rank symmetric matrix estimation problem. A similar idea was used in the context of phase retrieval in (Candès et al., 2013). Inspired by these works, we propose to convert the problem of estimating \( \lambda \) and \( P \) to that of estimating a low-rank symmetric tensor \( \mathcal{W} \in \mathbb{S}^{d^m} \). Combined with a symmetrization trick, this approach leads to an objective that is multi-convex, for both \( \mathcal{K} = \mathcal{A}^m \) and \( \mathcal{K} = \mathcal{H}^m \) (Section 5.2).

We begin by rewriting the kernel definitions using rank-one tensors. For \( \mathcal{H}^m(p, x) \), it is easy to see that

\[
\mathcal{H}^m(p, x) = \langle p^{\otimes m}, x^{\otimes m} \rangle.
\]

For \( \mathcal{A}^m(p, x) \), we need to ignore irrelevant monomials. For convenience, we introduce the following notation:

\[
\langle \mathcal{W}, \mathcal{X} \rangle := \sum_{j_1, \ldots, j_m} \mathcal{W}_{j_1, \ldots, j_m} \mathcal{X}_{j_1, \ldots, j_m} \quad \forall \mathcal{W}, \mathcal{X} \in \mathbb{S}^{d^m}.
\]

We can now concisely rewrite the ANOVA kernel as

\[
\mathcal{A}^m(p, x) = \langle p^{\otimes m}, x^{\otimes m} \rangle >.
\]

Our key insight is described in the following lemma.

**Lemma 5** Link between tensors and kernel expansions

Let \( \mathcal{W} \in \mathbb{S}^{d^m} \) have a symmetric outer product decomposition (Comon et al., 2008)

\[
\mathcal{W} = \sum_{s=1}^k \lambda_s \mathcal{P}_s^{\otimes m}.
\]

Let \( \lambda = [\lambda_1, \ldots, \lambda_k]^T \) and \( P = [p_1, \ldots, p_k] \). Then

\[
\langle \mathcal{W}, x^{\otimes m} \rangle = \langle \mathcal{H}^m(x; \lambda, P) \quad (8)
\]

\[
\langle \mathcal{W}, x^{\otimes m} \rangle > = \langle \mathcal{A}^m(x; \lambda, P) \quad (9)
\]

The result follows immediately from (5) and (6), and from the linearity of \( \langle \cdot, \cdot \rangle \) and \( \langle \cdot, \cdot \rangle > \). Given \( \mathcal{W} \in \mathbb{S}^{d^m} \), let us define the following objective functions

\[
L_{\mathcal{H}^m}(\mathcal{W}) := \sum_{i=1}^n \ell(y_i, \langle \mathcal{W}, x_i^{\otimes m} \rangle)
\]

\[
L_{\mathcal{A}^m}(\mathcal{W}) := \sum_{i=1}^n \ell(y_i, \langle \mathcal{W}, x_i^{\otimes m} \rangle >).
\]

If \( \mathcal{W} \) is decomposed as in (7), then from Lemma 5, we obtain \( L_K(\mathcal{W}) = D_K(\lambda, P) \) for \( K = \mathcal{H}^m \) or \( \mathcal{A}^m \). This
Polynomial Networks and Factorization Machines: New Insights and Efficient Training Algorithms
suggests that we can convert the problem of learning $\lambda$ and $P$ to that of learning a symmetric tensor $W$ of (symmetric) rank $k$. Thus, the problem of finding a small number of bases $p_1, \ldots, p_d$ and their associated weights $\lambda_1, \ldots, \lambda_k$ is converted to that of learning a low-rank symmetric tensor. Following (Candès et al., 2013), we call this approach lifted. Intuitively, we can think of $W$ as a tensor that contains the weights for predicting $y$ of monomials of degree $m$. For instance, when $m = 3$, $W_{i,j,k}$ is the weight corresponding to the monomial $x_ix_jx_k$.

5.2. Multi-convex formulation

Estimating a low-rank symmetric tensor $W \in S^{d^2}$ for arbitrary integer $m \geq 2$ is in itself a difficult non-convex problem. Nevertheless, based on a symmetrization trick, we can convert the problem to a multi-convex one, which we can easily minimize by alternating minimization. We first present our approach for the case $m = 2$ to give intuitions then explain how to extend it to $m \geq 3$.

Intuition with the second-order case. For the case $m = 2$, we need to estimate a low-rank symmetric matrix $W \in S^{d^2}$. Naively parameterizing $W = P \text{diag}(\lambda) P^T$ and solving for $\lambda$ and $P$ does not lead to a multi-convex formulation for the case $\mathcal{K} = \mathcal{H}^2$. This is due to the fact that $\langle P \text{diag}(\lambda) P^T, x^{\otimes 2} \rangle$ is quadratic in $P$. Our key idea is to parametrize $W = S(UV^T)$ where $U, V \in \mathbb{R}^{d \times r}$ and $S(M) := \frac{1}{2}(M + M^T)$ is the symmetrization of $M \in \mathbb{R}^{d^2}$. We then minimize $L_K(S(UV^T))$ w.r.t. $U, V$.

The main advantage is that both $\langle S(UV^T), \cdot \rangle$ and $\langle S(UV^T), \cdot \rangle^T$ are bi-linear in $U$ and $V$. This implies that $L_K(S(UV^T))$ is bi-convex in $U$ and $V$ and can therefore be efficiently minimized by alternating minimization. Once we obtained $W = S(UV^T)$, we can optionally compute its eigendecomposition $W = P \text{diag}(\lambda) P^T$, with $k = \text{rank}(W)$ and $r \leq k \leq 2r$, then apply (8) or (9) to obtain the model in kernel expansion form.

Extension to higher-order case. For $m \geq 3$, we now estimate a low-rank symmetric tensor $W = S(M) \in S^{d^m}$, where $M \in \mathbb{R}^{d^m}$ and $S(M)$ is the symmetrization of $M$ (cf. Appendix A.2). We decompose $M$ using $m$ matrices of size $d \times r$. Let us call these matrices $\{U^j\}_{j=1}^m$ and their columns $u^j_s = [u^j_{s,1}, \ldots, u^j_{s,r}]^T$. Then the decomposition of $M$ can be expressed as a sum of rank-one tensors

$$M = \sum_{s=1}^r u^s \otimes \cdots \otimes u^m. \quad (10)$$

Due to multi-linearity of (10) w.r.t. $U^1, \ldots, U^m$, the objective function $L_K$ is multi-convex in $U^1, \ldots, U^m$.

Computing predictions efficiently. When $K = \mathcal{H}^m$, predictions are computed by $\langle W, x^{\otimes m} \rangle$. To compute them efficiently, we use the following lemma.

Lemma 6 Symmetrization does not affect inner product

$$\langle S(M), X \rangle = \langle M, X \rangle \quad \forall M \in \mathbb{R}^{d^m}, X \in \mathbb{S}^{d^m}, m \geq 2. \quad (11)$$

Proof is given in Appendix A.2.

Using $x^{\otimes m} \in \mathbb{S}^{d^m}$, we have

$$\langle W, x^{\otimes m} \rangle = \langle M, x^{\otimes m} \rangle = \sum_{s=1}^r \prod_{i=1}^m \langle u^i_s, x \rangle. \quad (12)$$

As a result, we never need to explicitly compute the symmetrized tensor. For the case $\mathcal{K} = \mathcal{A}^2$, cf. Appendix D.3.

6. Regularization

In some applications, the number of bases or the rank constraint are not enough for obtaining good generalization performance and it is necessary to consider additional form of regularization. For the lifted objective with $\mathcal{K} = \mathcal{H}^2$ or $\mathcal{A}^2$, we use the typical Frobenius-norm regularization

$$\tilde{L}_K(U, V) := L_K(S(UV^T)) + \frac{\beta}{2} (\|U\|^2_F + \|V\|^2_F), \quad (12)$$

where $\beta > 0$ is a regularization hyper-parameter. For the direct objective, we introduce the new regularization

$$\tilde{D}_K(\lambda, P) := D_K(\lambda, P) + \beta \sum_{s=1}^d \|\lambda_s\| \|p_s\|^2. \quad (13)$$

This allows us to regularize $\lambda$ and $P$ with a single hyper-parameter. Let us define the following nuclear norm penalized objective:

$$\tilde{L}_K(M) := L_K(S(M)) + \beta \|M\|_* \quad (14)$$

We can show that (12), (13) and (14) are equivalent in the following sense.

Theorem 2 Equivalence of regularized problems

Let $\mathcal{K} = \mathcal{H}^2$ or $\mathcal{A}^2$, then

$$\min_{M \in \mathbb{R}^{d^m}} \tilde{L}_K(M) = \min_{U \in \mathbb{R}^{d \times r}, V \in \mathbb{R}^{r \times r}} \tilde{L}_K(U, V) = \min_{\lambda \in \mathbb{R}^k, P \in \mathbb{R}^{d \times k}} \tilde{D}_K(\lambda, P) \quad \text{where } \text{rank}(M^*) \leq r \text{ and } M^* \in \arg\min_{M \in \mathbb{R}^{d^m}} \tilde{L}_K(M).$$

Proof is given in Appendix C. Our proof relies on the variational form of the nuclear norm and is thus limited to $m = 2$. One of the key ingredients of the proof is to show that the minimizer of (14) is always a symmetric matrix. In addition to Theorem 2, from (Abernethy et al., 2009), we also know that every local minimum $U, V$ of (12) gives a global solution $UV^T$ of (14) provided that $\text{rank}(M^*) \leq r$. Proving a similar result for (13) is a future work. When $m \geq 3$, as used in our experiments, a squared Frobenius norm penalty on $P$ (direct objective) or on $\{U^j\}_{j=1}^m$ (lifted objective) works well in practice, although we lose the theoretical connection with the nuclear norm.
7. Coordinate descent algorithms

We now describe how to learn the model parameters by coordinate descent, which is a state-of-the-art learning-rate free solver for multi-convex problems (e.g., Yu et al. (2012)). In the following, we assume that $\ell$ is $\mu$-smooth.

**Direct objective with $K = \mathcal{A}^m$ for $m \in \{2, 3\}$.** First, we note that minimizing (13) w.r.t. $\lambda$ can be reduced to a standard $\ell_1$-regularized convex objective via a simple change of variable. Hence we focus on minimization w.r.t. $P$.

Let us denote the elements of $P$ by $p_{js}$. Then, our algorithm cyclically performs the following update for all $s \in [k]$ and $j \in [d]$:

$$p_{js} \leftarrow p_{js} - \eta^{-1} \left[ \sum_{i=1}^{n} \ell'(y_i, \hat{y}_i) \frac{\partial \hat{y}_i}{\partial p_{js}} + 2\beta \lambda_s |p_{js}| \right],$$

where $\eta := \mu \sum_{i=1}^{n} \left( \frac{\partial \hat{y}_i}{\partial p_{js}} \right)^2 + 2\beta |\lambda_s|$. Note that when $\ell$ is the squared loss, the above is equivalent to a Newton update and is the exact coordinate-wise minimizer.

The key challenge to use CD is computing $\frac{\partial \hat{y}_i}{\partial p_{js}}$. Note that minimizing (13) w.r.t. $\lambda$ is computing $\frac{\partial \hat{y}_i}{\partial p_{js}}$. Complete details are given in Appendix D.2.

**Lifted objective with $K = \mathcal{H}^m$.** Recall that we want to learn the matrices $\{U^t \}_{t=1}^m$, whose columns we denote by $u^t_s = [u^t_{s1}, \ldots, u^t_{sd}]^T$. Our algorithm cyclically performs the following update for all $t \in [m], s \in [r]$ and $j \in [d]$:

$$u^t_{js} \leftarrow u^t_{js} - \eta^{-1} \left[ \sum_{i=1}^{n} \ell'(y_i, \hat{y}_i) \frac{\partial \hat{y}_i}{\partial u^t_{js}} + \beta u^t_{js} \right],$$

where $\eta := \mu \sum_{i=1}^{n} \left( \frac{\partial \hat{y}_i}{\partial u^t_{js}} \right)^2 + \beta$. The main difficulty is computing $\frac{\partial \hat{y}_i}{\partial u^t_{js}}$. The main difficulty is computing $\frac{\partial \hat{y}_i}{\partial u^t_{js}} = \prod_{t' \neq t} (u^t_{s}, x_i) x_j$ efficiently. If for all $i \in [n]$ and for $t$ and $s$ fixed, we maintain $\xi_i := \prod_{t' \neq t} (u^t_{s}, x_i)$, then the cost of computing $\frac{\partial \hat{y}_i}{\partial u^t_{js}}$ is $O(1)$. Hence the cost of one epoch is $O(mnk^2(X))$, the same as SGD. Complete details are given in Appendix D.2.

**Convergence.** The above updates decrease the objective monotonically. Convergence to a stationary point is guaranteed following (Bertsekas, 1999, Proposition 2.7.1).

8. Inhomogeneous polynomial models

The algorithms presented so far are designed for homogeneous polynomial kernels $\mathcal{H}^m$ and $\mathcal{A}^m$. These kernels only use monomials of the same degree $m$. However, in many applications, we would like to use monomials of up to some degree. In this section, we propose a simple idea to do so using the algorithms presented so far, unmodified. Our key observation is that we can easily turn homogeneous polynomials into inhomogeneous ones by augmenting the dimensions of the training data with dummy features.

We begin by explaining how to learn inhomogeneous polynomial models using $\mathcal{H}^m$. Let us denote $\tilde{p}^T := [\gamma, p^T] \in \mathbb{R}^{d+1}$ and $\tilde{x}^T := [1, x^T] \in \mathbb{R}^{d+1}$. Then, we obtain

$$\mathcal{H}^m(\tilde{p}, \tilde{x}) = (\tilde{p}, \tilde{x})^m = (\gamma + (p, x))^m = \mathcal{P}^m(p, x).$$

Therefore, if we prepare the augmented training set $\tilde{x}_1, \ldots, \tilde{x}_n$, the problem of learning a model of the form $\sum_k \lambda_k \mathcal{P}^m(p, x)$ can be converted to that of learning a rank-$k$ symmetric tensor $\mathcal{W} \in \mathbb{S}^{(d+1)^m}$ using the method presented in Section 5. Note that the parameter $\gamma$ is automatically learned from data for each basis $p_s$.

Next, we explain how to learn inhomogeneous polynomial models using $\mathcal{A}^m$. Using Lemma 2, we immediately obtain for $1 \leq m \leq d$:

$$\mathcal{A}^m(\tilde{p}, \tilde{x}) = \mathcal{A}^m(p, x) + \gamma \mathcal{A}^{m-1}(p, x).$$

For instance, when $m = 2$, we obtain

$$\mathcal{A}^2(\tilde{p}, \tilde{x}) = \mathcal{A}^2(p, x) + \gamma \mathcal{A}^1(p, x) = \mathcal{A}^2(p, x) + \gamma (p, x).$$

Therefore, if we prepare the augmented training set $\tilde{x}_1, \ldots, \tilde{x}_n$, we can easily learn a combination of linear kernel and second-order ANOVA kernel using methods presented in Section 4 or Section 5. Note that (15) only states the relation between two ANOVA kernels of consecutive degrees. Fortunately, we can also apply (15) recursively. Namely, by adding $m - 1$ dummy features, we can sum the kernels from $\mathcal{A}^m$ down to $\mathcal{A}^1$ (i.e., linear kernel).

9. Experimental results

In this section, we present experimental results, focusing on regression tasks. Datasets are described in Appendix E. In all experiments, we set $\ell(y, \hat{y})$ to the squared loss.

9.1. Direct optimization: is it useful to fit $\lambda$?

As explained in Section 4, there is no benefit to fitting $\lambda$ when $m$ is odd, since $\mathcal{A}^m$ and $\mathcal{H}^m$ can absorb $\lambda$ into $P$. This is however not the case when $m$ is even: $\mathcal{A}^m$ and $\mathcal{H}^m$
9.2. Direct vs. lifted optimization

In this section, we compare the direct and lifted optimization approaches on high-dimensional data when $m = 2$. To compare the two approaches fairly, we propose the following initialization scheme. Recall that, at the end of the day, both approaches are essentially learning a low rank symmetric matrix: $W = S(UV^T)$ for lifted and $W = P \text{diag}(\lambda)P^T$ for direct optimization. This suggests that we can easily convert the matrices $U, V \in \mathbb{R}^{d \times r}$ used for initializing lifted optimization to $P \in \mathbb{R}^{d \times k}$ and $\lambda \in \mathbb{R}^{d \times k}$ by computing the (reduced) eigendecomposition of $S(UV^T)$. Note that because we solve the lifted optimization problem by coordinate descent, $UV^T$ is never symmetric and therefore the rank of $S(UV^T)$ is usually twice that of $UV^T$. Hence, in practice, we have that $r = k/2$. In our experiment, we compared four methods: lifted objective solved by CD, direct objective solved by CD, LBFGS and SGD. For lifted optimization, we initialized the elements of $U$ and $V$ by sampling from $N(0, 0.01)$. For direct optimization, we obtained $P$ and $\lambda$ as explained.

Results on the E2006-tfidf high-dimensional dataset are shown in Figure 2. For $\mathcal{K} = \mathcal{A}^2$, we find that Lifted (CD) and Direct (CD) have similar convergence speed and both outperform Direct (L-BFGS). For $\mathcal{K} = \mathcal{H}^2$, we find that Lifted (CD) outperforms both Direct (L-BFGS) and Direct (SGD). Note that we did not implement Direct (CD) for $\mathcal{K} = \mathcal{H}^2$ since the direct optimization problem is not coordinate-wise convex, as explained in Section 5.

9.3. Recommender system experiment

To confirm the ability of the proposed framework to infer the weights of unobserved feature interactions, we conducted experiments on Last.fm and Movielens 1M, two standard recommender system datasets. Following (Rendle, 2012), matrix factorization can be reduced to FMs by creating a dataset of $(x_i, y_i)$ pairs where $x_i$ contains the one-hot encoding of the user and item and $y_i$ is the corresponding rating (i.e., number of training instances equals number of ratings). We compared four models:

a) $\mathcal{K} = \mathcal{A}^2$ (augment): $\hat{y} = \hat{y}_{\mathcal{A}^2}(\tilde{x})$, with $\tilde{x}^T := [1, x^T]$

b) $\mathcal{K} = \mathcal{A}^2$ (linear combination): $\hat{y} = \langle w, x \rangle + \hat{y}_{\mathcal{A}^2}(x)$

c) $\mathcal{K} = \mathcal{H}^2$ (augment): $\hat{y} = \hat{y}_{\mathcal{H}^2}(\tilde{x})$ and

d) $\mathcal{K} = \mathcal{H}^2$ (linear combination): $\hat{y} = \langle w, x \rangle + \hat{y}_{\mathcal{H}^2}(x)$,

where $w \in \mathbb{R}^d$ is a vector of first-order weights, estimated from training data. Note that b) and d) are exactly the same as FMs and PNs, respectively. Results are shown in Figure 3. We see that $\mathcal{A}^2$ tends to outperform $\mathcal{H}^2$ on these tasks. We hypothesize that this the case because features are binary (cf., discussion in Section 10). We also see that simply augmenting the features as suggested in Section 8 is comparable or better than learning additional first-order feature weights, as done in FMs and PNs.
9.4. Low-budget non-linear regression experiment

In this experiment, we demonstrate the ability of the proposed framework to reach good regression performance with a small number of bases $k$. We compared:

a) Proposed with $\mathcal{K} = \mathcal{H}^3$ (with augmented features),
b) Proposed with $\mathcal{K} = \mathcal{A}^3$ (with augmented features),
c) Nyström method with $\mathcal{K} = \mathcal{P}_k$ and
d) Random Selection: choose $p_1, \ldots, p_k$ uniformly at random from training set and use $\mathcal{K} = \mathcal{P}_k$.

For a) and b) we used the lifted approach. For fair comparison in terms of model size (number of floats used), we set $r = k/3$. Results on the abalone, cadata and cupsmall datasets are shown in Figure 4. We see that i) the proposed framework reaches the same performance as kernel ridge regression with much fewer bases than other methods and ii) $\mathcal{H}^3$ tends to outperform $\mathcal{A}^3$ on these tasks. Similar trends were observed when using $\mathcal{K} = \mathcal{H}^2$ or $\mathcal{A}^2$.

10. Discussion

Ability to infer weights of unobserved interactions. In our view, one of the strengths of PNs and FMs is their ability to infer the weights of unobserved feature interactions, unlike traditional kernel methods. To see why, recall that in kernel methods, predictions are computed by $\hat{y} = \sum_{i=1}^{n} \alpha_i \mathcal{K}(x_i, x)$. When $\mathcal{K} = \mathcal{H}^m$ or $\mathcal{A}^m$, by Lemma 5, this is equivalent to $\hat{y} = \langle \tilde{W}, x^{\otimes m} \rangle$ or $\langle \tilde{W}, x^{\otimes m} \rangle$. Thus, in kernel methods, the weight associated with $x_{j_1} \ldots x_{j_m}$ can be written as a linear combination of the training data’s monomials:

$$\tilde{W}_{j_1, \ldots, j_m} = \sum_{i=1}^{n} \alpha_i x_{j_1i} \ldots x_{j_mi}.$$  

Assuming binary features, the weights of monomials that were never observed in the training set are zero. In contrast, in PNs and FMs, we have $\mathcal{W} = \sum_{s=1}^{k} \lambda_s p_s^{\otimes m}$ and therefore the weight associated with $x_{j_1} \ldots x_{j_m}$ becomes

$$\mathcal{W}_{j_1, \ldots, j_m} = \sum_{s=1}^{k} \lambda_s p_{j_1s} \ldots p_{j_ms}.$$  

Because parameters are shared across monomials, PNs and FMs are able to interpolate the weights of monomials that were never observed in the training set. This is the key property which makes it possible to use them on recommender system tasks. In future work, we plan to apply PNs and FMs to biological data, where this property should be
Polynomial Networks and Factorization Machines: New Insights and Efficient Training Algorithms

(a) abalone
(b) cpusmall
(c) cadata

Figure 4. Regression performance as a function of the number of bases. The metric used is the coefficient of determination on the test set (higher is better). Regularization parameter was selected from 10 log-spaced values in $[10^{-4}, 10^{6}]$ by 5-fold cross-validation.

very useful, e.g., for inferring higher-order interactions between genes.

ANOVA kernel vs. polynomial kernel. One of the key properties of the ANOVA kernel $\mathcal{A}^m(p, x)$ is multilinearity w.r.t. elements of $p$ (Lemma 2). This is the key difference with $\mathcal{H}^m(p, x)$ which makes the direct optimization objective multi-convex when $\mathcal{K} = \mathcal{A}^m$ (Theorem 1). However, because we need to ignore irrelevant monomials, computing the kernel and its gradient is more challenging. Deriving efficient training algorithms for arbitrary $m \in \mathbb{N}$ is an important future work.

In our experiments in Section 9.1, we showed that fixing $\lambda = 1$ works relatively well when $\mathcal{K} = \mathcal{A}^2$. To see intuitively why this is the case, note that fixing $\lambda = 1$ is equivalent to constraining the weight matrix $W$ to be positive semidefinite, i.e., $\exists P$ s.t. $W = PP^T$. Next, observe that we can rewrite the prediction function as

$$\hat{y}_{\mathcal{A}^2}(x; 1, P) = \langle PP^T, x^{\otimes 2} \rangle = \langle U(PP^T), x^{\otimes 2} \rangle,$$

where $U(M)$ is a mask which sets diagonal and lower-diagonal elements of $M$ to zero. We therefore see that when using $\mathcal{K} = \mathcal{A}^2$, we are learning a strictly upper-triangular matrix, parametrized by $PP^T$. Importantly, the matrix $U(PP^T)$ is not positive semidefinite. This is what gives the model some degree of freedom, even though $PP^T$ is positive semidefinite. In contrast, when using $\mathcal{K} = \mathcal{H}^2$, if we fix $\lambda = 1$, then we have that

$$\hat{y}_{\mathcal{H}^2}(x; 1, P) = \langle PP^T, x^{\otimes 2} \rangle = x^T PP^T x \geq 0$$

and therefore the model is unable to predict negative values.

Empirically, we showed in Section 9.4 that $\mathcal{H}^m$ outperforms $\mathcal{A}^m$ for low-budget non-linear regression. In contrast, we showed in Section 9.3 that $\mathcal{A}^m$ outperforms $\mathcal{H}^m$ for recommender systems. The main difference between the two experiments is the nature of the features used: continuous for the former and binary for the latter. For binary features, squared features $x_1^2, \ldots, x_d^2$ are redundant with $x_1, \ldots, x_d$ and are therefore not expected to help improve accuracy. On the contrary, they might introduce bias towards first-order features. We hypothesize that the ANOVA kernel is in general a better choice for binary features, although this needs to be verified by more experiments, for instance on natural language processing (NLP) tasks.

Direct vs. lifted optimization. The main advantage of direct optimization is that we only need to estimate $\lambda \in \mathbb{R}^k$ and $P \in \mathbb{R}^{d \times k}$ and therefore the number of parameters to estimate is independent of the degree $m$. Unfortunately, the approach is neither convex nor multi-convex when using $\mathcal{K} = \mathcal{H}^m$. In addition, the regularized objective (13) is non-smooth w.r.t. $\lambda$. In Section 5, we proposed to reformulate the problem as one of low-rank symmetric tensor estimation and used a symmetrization trick to obtain a multi-convex smooth objective function. Because this objective involves the estimation of $m$ matrices of size $d \times r$, we need to set $r = k/m$ for fair comparison with the direct objective in terms of model size. When $\mathcal{K} = \mathcal{A}^m$, we showed that the direct objective is readily multi-convex. However, an advantage of our lifted objective when $\mathcal{K} = \mathcal{A}^m$ is that it is convex w.r.t. larger block of variables than the direct objective.

11. Conclusion

In this paper, we revisited polynomial networks (Livni et al., 2014) and factorization machines (Rendle, 2010; 2012) from a unified perspective. We proposed direct and lifted optimization approaches and showed their equivalence in the regularized case for $m = 2$. With respect to PNs, we proposed the first CD solver with support for arbitrary integer $m \geq 2$. With respect to FMs, we made several novel contributions including making a connection with the ANOVA kernel, proving important properties of the objective function and deriving the first CD solver for third-order FMs. Empirically, we showed that the proposed algorithms achieve excellent performance on non-linear regression and recommender system tasks.
Acknowledgments

This work was partially conducted as part of “Research and Development on Fundamental and Applied Technologies for Social Big Data”, commissioned by the National Institute of Information and Communications Technology (NICT), Japan. We also thank Vlad Niculae, Olivier Grisel, Fabian Pedregosa and Joseph Salmon for their valuable comments.

References


