Shallow Learning with Kernels for Dictionary-Free Magnetic Resonance Fingerprinting

Gopal Nataraj*, Mingjie Gao*, Jakob Assländer†, Clayton Scott*, & Jeffrey A. Fessler*

ISM RM Workshop on Magnetic Resonance Fingerprinting

*Dept. of Electrical Engineering and Computer Science, University of Michigan
†Center for Biomedical Imaging, NYU School of Medicine
Problem Statement

Given: at every voxel, measurement vector $y = s(x) + \epsilon$

MRF “component” images (more later...)
**Problem Statement**

**Given:** at every voxel, measurement vector $\mathbf{y} = \mathbf{s}(\mathbf{x}) + \epsilon$

MRF “component” images (more later...)

**Task:** design fast voxel-by-voxel estimator $\hat{\mathbf{x}}(\cdot)$ that scales well with #unknowns per voxel, $L$
Idea: “learn” separate scalar estimators $\hat{x}_1(y), \ldots, \hat{x}_L(y)$ from simulated training data
Idea: “learn” separate scalar estimators $\hat{x}_1(y), \ldots, \hat{x}_L(y)$ from simulated training data

Deep Learning
- promising for QMRI [Cohen et al., 2017, Virtue et al., 2017]
- needs many training points to avoid overfitting
- trained via non-convex optimization
- limited theoretical basis
**Machine Learning at Different “Depths” for QMRI**

**Idea:** “learn” separate scalar estimators $\hat{x}_1(y), \ldots, \hat{x}_L(y)$ from simulated training data

**Shallow Learning**
- simpler structure needs fewer training points
- fast training via convex optimization

**Deep Learning**
- promising for QMRI [Cohen et al., 2017, Virtue et al., 2017]
- needs many training points to avoid overfitting
- trained via non-convex optimization
- limited theoretical basis
Idea: “learn” separate scalar estimators $\hat{x}_1(y), \ldots, \hat{x}_L(y)$ from simulated training data

- sample $(x_1, \epsilon_1), \ldots, (x_N, \epsilon_N)$ and simulate $y_1, \ldots, y_N$ via signal model $s$
Idea: “learn” separate scalar estimators $\hat{x}_1(y), \ldots, \hat{x}_L(y)$ from simulated training data

- sample $(x_1, \epsilon_1), \ldots, (x_N, \epsilon_N)$ and simulate $y_1, \ldots, y_N$ via signal model $s$
- design nonlinear functions $\hat{x}_l(\cdot) := \hat{g}_l(\cdot) + \hat{b}_l$ that seek to map each $y_n$ to $x_{l,n}$:

$$\left(\hat{g}_l, \hat{b}_l\right) \in \left\{ \arg \min_{\hat{g}_l, \hat{b}_l \in \mathbb{R}} \frac{1}{N} \sum_{n=1}^{N} \left(\hat{g}_l(y_n) + \hat{b}_l - x_{l,n}\right)^2 \right\}$$
Idea: “learn” separate scalar estimators $\hat{x}_1(y), \ldots, \hat{x}_L(y)$ from simulated training data

- sample $(x_1, \epsilon_1), \ldots, (x_N, \epsilon_N)$ and simulate $y_1, \ldots, y_N$ via signal model $s$
- design nonlinear functions $\hat{x}_l(\cdot) := \hat{g}_l(\cdot) + \hat{b}_l$ that seek to map each $y_n$ to $x_{l,n}$:

$$\left(\hat{g}_l, \hat{b}_l\right) \in \left\{ \arg\min_{\hat{g}_l, \hat{b}_l \in \mathbb{R}^N} \frac{1}{N} \sum_{n=1}^{N} (g_l(y_n) + b_l - x_{l,n})^2 \right\} \text{ ill-posed!}$$
Idea: “learn” separate scalar estimators \( \hat{x}_1(y), \ldots, \hat{x}_L(y) \) from simulated training data

- sample \((x_1, \epsilon_1), \ldots, (x_N, \epsilon_N)\) and simulate \(y_1, \ldots, y_N\) via signal model \(s\)
- design nonlinear functions \(\hat{x}_l(\cdot) := \hat{g}_l(\cdot) + \hat{b}_l\) that seek to map each \(y_n\) to \(x_{l,n}:\)

\[
\left(\hat{g}_l, \hat{b}_l\right) \in \left\{ \arg \min_{g_l \in G} \frac{1}{N} \sum_{n=1}^{N} (g_l(y_n) + b_l - x_{l,n})^2 + \rho_l \|g_l\|_G^2 \right\}
\]  

(1)

Solution: Parameter Estimation via Regression with Kernels (PERK)

[Nataraj et al., 2017b, arXiv:1710.02441]

- restrict optimization to a certain rich function space \(G\) with kernel \(k\)
- optimal \(\hat{g}_l \in G\) takes form \(\hat{g}_l(\cdot) = \sum_{n=1}^{N} \hat{a}_{l,n}k(\cdot, y_n)\)  
  [Schölkopf et al., 2001]
Shallow Learning with Kernels for QMRI

**Idea:** “learn” separate scalar estimators $\hat{x}_1(y), \ldots, \hat{x}_L(y)$ from simulated training data

- sample $(x_1, \epsilon_1), \ldots, (x_N, \epsilon_N)$ and simulate $y_1, \ldots, y_N$ via signal model $s$
- design *nonlinear* functions $\hat{x}_l(\cdot) := \hat{g}_l(\cdot) + \hat{b}_l$ that seek to map each $y_n$ to $x_{l,n}$:

$$
\left(\hat{g}_l, \hat{b}_l\right) \in \left\{ \operatorname{arg} \min_{g_l \in G} \frac{1}{N}\sum_{n=1}^{N} \left( g_l(y_n) + b_l - x_{l,n} \right)^2 + \rho_l \| g_l \|_G^2 \right\} \quad (1)
$$

**Solution:** Parameter Estimation via Regression with Kernels (PERK)

- restrict optimization to a certain rich function space $G$ with kernel $k$
- optimal $\hat{g}_l \in G$ takes form $\hat{g}_l(\cdot) = \sum_{n=1}^{N} \hat{a}_{l,n} k(\cdot, y_n)$

[Schölkopf et al., 2001]

**Fast, simple implementation:** nonlinear *lifting* + high-dimensional linear regression

[Nataraj et al., 2017b, arXiv:1710.02441]
To control lifting dimension, desirable for $y$ to be low-dimensional
PERK for Magnetic Resonance Fingerprinting (MRF)

To control lifting dimension, desirable for $y$ to be low-dimensional.

Data-sharing across flips; gridding; FFT; PCA

[Assländer et al., 2017]
PERK for Magnetic Resonance Fingerprinting (MRF)

To control lifting dimension, desirable for $y$ to be low-dimensional

$$k_y \quad \rightarrow \quad k_x \quad \text{flip } 1$$

$$\vdots \quad \vdots \quad \vdots$$

$$k_y \quad \rightarrow \quad k_x \quad \text{flip } 840$$

[Assländer et al., 2017]

Data-sharing across flips; gridding; FFT; PCA

$$V \in \mathbb{C}^{840 \times 6}$$

$$\min_{Y} \| k - \mathcal{A}(YV^H) \|_2^2$$

$$Y \in \mathbb{C}^{n_{\text{voxels}} \times 6}$$

[Image showing data over time and associated with the mathematical expressions.]
PERK for Magnetic Resonance Fingerprinting (MRF)

To control lifting dimension, desirable for $\mathbf{y}$ to be low-dimensional.

\[
\min_{\mathbf{Y}} \| \mathbf{k} - \mathcal{A}(\mathbf{YV}^H) \|_2^2
\]

\[\mathbf{V} \in \mathbb{C}^{840 \times 6}\]

\[\mathbf{Y} \in \mathbb{C}^{n_{\text{voxels}} \times 6}\]

[Assländer et al., 2017]
PERK for Magnetic Resonance Fingerprinting (MRF)

To control lifting dimension, desirable for $y$ to be low-dimensional

\[ k_y \rightarrow k_x \quad \text{flip 1} \]

\[ \vdots \quad \vdots \]

\[ k_y \rightarrow k_x \quad \text{flip 840} \]

\[ \left. \begin{array}{c}
\text{data-sharing across flips;} \\
\text{gridding; FFT; PCA}
\end{array} \right\} \]

\[ Y \in \mathbb{C}^{840 \times 6} \]

\[ \min_Y \| k - \mathcal{A}(Y V^H) \|_2^2 \]

\[ Y \in \mathbb{C}^{n_{\text{voxels}} \times 6} \]

[Assländer et al., 2017]
In vivo results

Dictionary-based Grid Search

Dictionary-Free PERK

T1

T2

50 80 110 140 170 200 ms

600 800 1000 1200 1400 1600 1800 2000 ms

flip 1

flip 840

$k_y$ $k_x$

$k_y$ $k_x$
In vivo results

\[ k_y \quad \rightarrow \quad k_x \quad \text{flip 1} \]

\[ k_y \quad \rightarrow \quad k_x \quad \text{flip 840} \]

**Dictionary-based Grid Search**

**Dictionary-Free PERK**

T1

T2
In vivo results

$T_2$ (ms)

50 80 110 140 170 200

$T_1$ (ms)

50 80 110 140 170 200

ms

50 100 150 200 250 300 400

Dictionary-based Grid Search

Dictionary-Free PERK

$T_1$ slice 5$

\sim 1400s$

4s train; 0.2s/slice test

$\sim 10s$ test

$\sim 6$
In vivo results

[Diagram showing $k_x$ and $k_y$ with arrows indicating $k_x$ flip 1 and $k_x$ flip 840]

Dictionary-based Grid Search

Dictionary-Free PERK

T1

T2

28s/slice 4s train; 0.2s/slice test
In vivo results

$\begin{align*}
k_y \\ &\downarrow \\
&k_x \\
&\rightarrow \text{flip 1} \\
\vdots \\

k_y \\ &\downarrow \\
&k_x \\
&\rightarrow \text{flip 840}
\end{align*}$

50 slices

$\sim 1400s$

$28s/\text{slice}$

$4s$ train; $0.2s/\text{slice}$ test

$4s$ train; $\sim 10s$ test
Summary

Contributions:

- **PERK**: fast, dictionary-free ML method for QMRI [Nataraj et al., 2017b]
- demonstrated PERK for *in vivo* MRF $T_1, T_2$ estimation
Summary

Contributions:

- **PERK**: fast, dictionary-free ML method for QMRI [Nataraj et al., 2017b]
- demonstrated PERK for *in vivo* MRF $T_1$, $T_2$ estimation

Future Work:

- QMRI problems involving more unknowns for which we expect orders-of-magnitude computational gains [Nataraj et al., 2017a, #5076]
Summary

Contributions:

- **PERK**: fast, dictionary-free ML method for QMRI [Nataraj et al., 2017b]
- demonstrated PERK for *in vivo* MRF $T_1$, $T_2$ estimation

Future Work:

- QMRI problems involving more unknowns for which we expect *orders-of-magnitude* computational gains [Nataraj et al., 2017a, #5076]
- comparison with other ML methods, *e.g.* deep learning
arxiv 1703.00481.


Better than real: Complex-valued neural nets for MRI fingerprinting.
To appear.
PERK solution

Closed-form solution for each $l \in \{1, \ldots, L\}$:

$$\hat{x}_l(\cdot) = x_l^T \left( \frac{1}{N} 1_N + M \left( MKM + N \rho_l 1_N \right)^{-1} \left( k(\cdot) - \frac{1}{N} K 1_N \right) \right)$$  \hspace{1cm} (2)

- $x_l := [x_{l,1}, \ldots, x_{l,N}]^T$ \hspace{1cm} training point regressands
- $K := \begin{bmatrix} k(y_1, y_1) & \cdots & k(y_1, y_N) \\ \vdots & \ddots & \vdots \\ k(y_N, y_1) & \cdots & k(y_N, y_N) \end{bmatrix}$ \hspace{1cm} Gram matrix
- $M := I_N - \frac{1}{N} 1_N 1_N^T$ \hspace{1cm} de-meaning operator
- $k(\cdot) := [k(\cdot, y_1), \ldots, k(\cdot, y_N)]^T$ \hspace{1cm} nonlinear kernel embedding

Can we scale computation with $L$ more gracefully?

- Yes, in fact (2) separable in $l \in \{1, \ldots, L\}$ by construction
PERK solution

Closed-form solution for each $l \in \{1, \ldots, L\}$:

$$\hat{x}_l(\cdot) = x_l^T \left( \frac{1}{N} \mathbf{1}_N + M(MKM + N\rho_l \mathbf{1}_N)^{-1} \left( k(\cdot) - \frac{1}{N} K \mathbf{1}_N \right) \right)$$  \hspace{1cm} (2)

- $x_l := [x_{l,1}, \ldots, x_{l,N}]^T$ \hspace{1cm} training point regressands
- $K := \begin{bmatrix} k(y_1, y_1) & \cdots & k(y_1, y_N) \\ \vdots & \ddots & \vdots \\ k(y_N, y_1) & \cdots & k(y_N, y_N) \end{bmatrix}$ \hspace{1cm} Gram matrix
- $M := \mathbf{I}_N - \frac{1}{N} \mathbf{1}_N \mathbf{1}_N^T$ \hspace{1cm} de-meaning operator
- $k(\cdot) := [k(\cdot, y_1), \ldots, k(\cdot, y_N)]^T$ \hspace{1cm} nonlinear kernel embedding

Can we scale computation with $L$ more gracefully?

- Yes, in fact (2) separable in $l \in \{1, \ldots, L\}$ by construction
- However, explicitly computing $K$ may be undesirable...
Suppose there exists “approximate feature mapping” \( \tilde{z} : \mathcal{Y} \mapsto \mathbb{R}^Z \)
such that \( \tilde{Z} := [\tilde{z}(y_1), \ldots, \tilde{z}(y_N)] \) has for \( \dim(\mathcal{Y}) \ll Z \ll N \)

\[
K \approx \tilde{Z}^T \tilde{Z}.
\] (3)

Plugging (3) into PERK solution (2) and rearranging gives

\[
\hat{x}_l(\cdot) \approx \frac{1}{N} x_l^T 1_N + \frac{1}{N} x_l^T M \tilde{Z}^T \left( \frac{1}{N} \tilde{Z} M \tilde{Z}^T + \rho I_Z \right)^{-1} \left( \tilde{z}(\cdot) - \frac{1}{N} \tilde{Z} 1_N \right)
\]
Suppose there exists “approximate feature mapping” \( \tilde{z} : \mathcal{Y} \mapsto \mathbb{R}^Z \) such that \( \tilde{Z} := [\tilde{z}(y_1), \ldots, \tilde{z}(y_N)] \) has for \( \dim(\mathcal{Y}) \ll Z \ll N \)

\[
K \approx \tilde{Z}^T \tilde{Z}.
\]

(3)

Plugging (3) into KRR solution (2) and rearranging gives

\[
\hat{x}_l(\cdot) \approx \hat{m}_{xl} + \hat{c}_{x_l}^T \left( \hat{C}_{\tilde{z} \tilde{z}} + \rho_l I_Z \right)^{-1} (\tilde{z}(\cdot) - \hat{m}_\tilde{z})
\]

(4)

which is regularized (“ridge”) \( Z \)-dimensional affine regression!

Does such a \( \tilde{z} \) exist and work well in practice?

- Yes, e.g. for kernels of form \( k(y, y') \equiv k(y - y') \) [Rahimi and Recht, 2007]
- In such cases, can reduce from \( \sim N^2 \) to \( \sim NZ \) computations