Leveraging Signal Structure for Efficient and Adaptive Machine Learning

John Lipor

January 24, 2020
What is machine learning?
Machine learning refers to a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data.\(^1\)

\(^1\)Machine Learning, A Probabilistic Perspective, Kevin P. Murphy.
Machine learning refers to a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data\(^1\).

\(^1\)Machine Learning, A Probabilistic Perspective, Kevin P. Murphy.
A brief history lesson

- Classical DSP: Signals in a Fourier basis
A brief history lesson

• Classical DSP: Signals in a Fourier basis
• Next up: Signals in other bases
A brief history lesson

- Classical DSP: Signals in a Fourier basis
- Next up: Signals in other bases
A brief history lesson

- Classical DSP: Signals in a Fourier basis
- Next up: Signals in other bases
- Most recently: Approximate signal recovery

Figure 1: source: Baraniuk, “Compressive Sensing.”
A brief history lesson

- Classical DSP: Signals in a Fourier basis
- Next up: Signals in other bases
- Most recently: Approximate signal recovery

**Trend:** Move from signal *reconstruction* $y = Ax$ to *optimization* $y = \text{arg min}_x \|y - Ax\| + \lambda \Omega(x)$. In both cases, leverage *structure* in the signals.

**Figure 1:** source: Baraniuk, “Compressive Sensing.”
Machine learning is “just” optimization

Least squares:

\[ y_i \in \mathbb{R} \text{ at location } x_i, i = 1, \ldots, N \]

- call \( x_i \) a “feature vector”
- call \( y_i \) a “label”
Machine learning is “just” optimization

Least squares:
- measure $y_i \in \mathbb{R}$ at location
  $x_i \in \mathbb{R}^D$, $i = 1, \ldots, N$
Machine learning is “just” optimization

Least squares:
- measure $y_i \in \mathbb{R}$ at location $x_i \in \mathbb{R}^D$, $i = 1, \ldots, N$
- call $x_i$ a “feature vector”
Machine learning is “just” optimization

Least squares:
- measure $y_i \in \mathbb{R}$ at location $x_i \in \mathbb{R}^D$, $i = 1, \ldots, N$
- call $x_i$ a “feature vector”
- call $y_i$ a “label”
Least squares:

- measure $y_i \in \mathbb{R}$ at location $x_i \in \mathbb{R}^D$, $i = 1, \ldots, N$
- call $x_i$ a “feature vector”
- call $y_i$ a “label”

Approximate $y_i \approx f(x_i)$ by assuming $f$ is linear, i.e., $y_i = w^T x_i$. 

Machine learning is “just” optimization
Least squares:

- measure $y_i \in \mathbb{R}$ at location $x_i \in \mathbb{R}^D, i = 1, \cdots, N$
- call $x_i$ a “feature vector”
- call $y_i$ a “label”

Solve the linear least squares problem:

$$ w^* = \arg \min_w \|y - Xw\|_2^2 $$
Machine learning is “just” optimization

Least squares:

- measure $y_i \in \mathbb{R}$ at location $x_i \in \mathbb{R}^D$, $i = 1, \ldots, N$
- call $x_i$ a “feature vector”
- call $y_i$ a “label”

Another view of least squares:

$$w^* = \arg \min_w \frac{1}{N} \sum_{i=1}^{N} (y_i - w^T x_i)^2$$
Machine learning is “just” optimization

Machine learning generalizes this beyond regression and beyond linear functions:

\[ f^* = \arg \min_f \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda \Omega(f) \]
Machine learning is “just” optimization

Machine learning generalizes this beyond regression and beyond linear functions:

$$f^* = \arg \min_f \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda \Omega(f)$$

\[
\begin{align*}
L(y, f(x)) &= \begin{cases} 
0 & \text{if } y = f(x) \\
(y - f(x))^2 & \text{otherwise}
\end{cases} \\
\Omega(f) &= \begin{cases} 
0 & \text{if } f(x) \text{ is a linear function}
\text{otherwise}
\end{cases}
\end{align*}
\]
Machine learning generalizes this beyond regression and beyond linear functions:

\[ f^* = \arg \min_{f} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda \Omega(f) \]

**Takeaway:** Much of ML can be boiled down to solving optimization problems.
Machine learning is “just”
Other perspectives

Machine learning is “just”

- function approximation
Other perspectives

Machine learning is “just”

- function approximation
- statistics
Other perspectives

Machine learning is “just”

- function approximation
- statistics
- neural networks(?)

hype!!
Other perspectives

Machine learning is “just”

- function approximation
- statistics
- neural networks(?)
- hype!!
Machine learning is “just”

- function approximation
- statistics
- neural networks(?)
- hype!!
Other perspectives

Machine learning is “just”

- function approximation
- statistics
- neural networks(?)
- hype!!

My research: A mix of tools from optimization, function approximation, statistics, signal processing.
Topic 1: Subspace Clustering
**Premise:** High-dimensional data often dominated by low-dimensional structure.
**Premise:** High-dimensional data often dominated by low-dimensional structure.
Premise: High-dimensional data often dominated by low-dimensional structure.
**Premise:** High-dimensional data often dominated by low-dimensional structure.
Subspace clustering: Unsupervised learning of $K$ unknown subspaces that best represent our data
A simple approach to subspace clustering

- Initialize $K$ random subspaces (one for each class) with bases $U_1, \ldots, U_K$.
A simple approach to subspace clustering

- Initialize $K$ random subspaces (one for each class) with bases $U_1, \ldots, U_K$
- Cluster points based on nearest subspace

$$y_i = \arg \min_k \|U_k^T x_i\|_2$$
A simple approach to subspace clustering

- Initialize $K$ random subspaces (one for each class) with bases $U_1, \ldots, U_K$
- Cluster points based on nearest subspace

$$y_i = \arg \min_k ||U_k^T x_i||_2$$

- Estimate subspace bases using points in each class

$$U_k = \text{PCA}(\{x_i : y_i = k\})$$
A simple approach to subspace clustering

- Initialize $K$ random subspaces (one for each class) with bases $U_1, \ldots, U_K$

- Cluster points based on nearest subspace

\[ y_i = \arg \min_k \|U_k^T x_i\|_2 \]

- Estimate subspace bases using points in each class

\[ U_k = \text{PCA}(\{x_i : y_i = k\}) \]

This algorithm is called $K$-subspaces (KSS) and was thought to have poor empirical performance and no theoretical guarantees.
Problems with subspace clustering

Drawbacks to KSS:

- Sensitive to initialization
- PCA (subspace estimation) is sensitive to outliers

General problems in subspace clustering:

- What to do with side information?
- How to know if your output clustering is “good”?
Mainly focused on theoretical characterization of existing algorithms.
Contribution 1: Robust PCA in subspace clustering

Mainly focused on theoretical characterization of existing algorithms. Ugly results like this:

\[ \frac{N_1}{r_1} \left(1 - \cos^2(\theta^*)\right) - 1 - \frac{1}{r_1} > \]

\[ \max_{k \in \{2, \ldots, K\}} \left[ \frac{N_k}{r_k} \left(1 - \cos^2(\theta_{max}^{(1,k)})\right) - \frac{1}{r_k} \right. \]

\[ + \sum_{l \neq 1, k} \frac{N_l}{r_l} \left(\cos^2(\theta_{min}^{(k,l)}) - \cos^2(\theta_{max}^{(1,l)})\right) \]

\[ + \eta_1 \left(1 + \cos^2(\theta_{min}^{(1,k)})\right) + \eta_k \left(1 + \cos^2(\theta_{max}^{(1,k)})\right) \]

\[ + \sum_{l \neq 1, k} \eta_l \left(\cos^2(\theta_{min}^{(k,l)}) + \cos^2(\theta_{max}^{(k,l)})\right) \]  \( (6) \)

where

\[ \theta^* = \min_{k \in \{2, \ldots, K\}} \theta_{min}^{(1,k)} \]

and \( \eta_l = \max \left(\frac{4}{3} \log \frac{2r_l}{\delta}, \sqrt{\frac{4N_l}{r_l}} \log \frac{2r_l}{\delta}\right) \) for \( l \in \{1, \ldots, K\} \).
Contribution 1: Robust PCA in subspace clustering

Mainly focused on theoretical characterization of existing algorithms. But nice empirical results like these:
Contribution 2: Ensembles of KSS

Instead of using a single random initialization, combine many ($\approx 1000$) to get better performance.

Similarity between points after one run:
Contribution 2: Ensembles of KSS

Instead of using a single random initialization, combine many ($\approx 1000$) to get better performance.

Similarity after combining many initializations:
Contribution 3: Incorporating side information

Are these the same person?
Contribution 3: Incorporating side information

Yale, K = 5

Yale, K = 10

MNIST, K = 5

MNIST, K = 10

number of pairwise comparisons

missclassification %

SUPERPAC
URASC
Random
Oracle UoS
Topic 2: Spatial Sampling
A Motivating Problem
**Goal:** Determine the spatial extent of hypoxic (low-oxygen) regions in Lake Erie
Hypoxia detection in Lake Erie

**Goal:** Determine the spatial extent of hypoxic (low-oxygen) regions in Lake Erie

- Sample using a mobile/autonomous vehicle
- Minimize total sampling time, a function of number of samples and travel time
- Other applications in wildfire tracking, geothermal energy favorability, document classification
Spatial Sampling as Active Learning
Two-dimensional problem hard to approach directly; instead, sample along series of transects.
One-dimensional reformulation

Two-dimensional problem hard to approach directly; instead, sample along series of transects.
One-dimensional reformulation

Two-dimensional problem hard to approach directly; instead, sample along series of transects.
Two-dimensional problem hard to approach directly; instead, sample along series of transects.
One-dimensional reformulation

Two-dimensional problem hard to approach directly; instead, sample along series of transects.
One-dimensional reformulation

Two-dimensional problem hard to approach directly; instead, sample along series of transects.
One-dimensional reformulation

Think about sampling along each strip:

- initial location: $X_0 = 0$
- change point: $\theta$
- measurements:
  $$Y_n = 1 \{ X_n \leq \theta \}$$
- measurement cost: $T_s$
- travel cost: $T_t$

Goal:

$$\min_{X_1, \ldots, X_N \in [0,1]} T_s N + T_t \sum_{n=1}^N |X_n - X_{n-1}|$$

s. t. $$\left| \theta - \hat{\theta}_N \right| \leq \varepsilon$$
Reformulating the one-dimensional reformulation

Think about sampling along each strip:

- initial location: $X_0 = 0$
- change point: $\theta$
- measurements: $Y_n = 1 \{ X_n \leq \theta \}$
- measurement cost: $T_s$
- travel cost: $T_t$

Goal:

$$\min_{X_1, \ldots, X_N \in [0,1]} T_s N + T_t \sum_{n=1}^{N} |X_n - X_{n-1}|$$

s. t. $$|\theta - \hat{\theta}_N| \leq \varepsilon$$
Reformulating the one-dimensional reformulation

Key idea: Fix number of samples and think in terms of sampling fractions

- sampling fractions: \( x_1, x_2, \ldots, x_N \)
- entropy after \( N \) measurements: \( H_N \)
- distance after \( N \) measurements: \( D_N \)
- tuning parameter: \( \lambda \in [0, 2] \)

Goal:

\[
\min_{x_1, \ldots, x_N \in [0, 1]} \mathbb{E} \left[ e^{H_N} + \lambda D_N \right]
\]
Benefits of a fixed horizon

Fixing $N$ allows us to determine the optimal sampling fractions in closed form.
Benefits of a fixed horizon

Theorem

Let $\lambda \in [0, 2]$ and assume the change point has distribution $\theta \sim \text{Unif}([0, 1])$. Further, assume the $N$ measurements are defined via $N$ fractions $x_1, \ldots, x_N$ denoting the proportion of the current hypothesis space to sample. Define

$$\xi_i = x_i^2 + (1 - x_i)^2, \quad i = 1, \ldots, N.$$  

Then,

$$\mathbb{E} \left[ e^{HN} + \lambda D_N \right] = \prod_{i=1}^{N} \xi_i + \lambda \sum_{i=1}^{N} x_i \prod_{j=0}^{i-1} \xi_j.$$
Theorem

Under the same conditions as the previous theorem, the optimal sampling fractions are of the form

\[ x_k = \frac{1}{2} - \lambda \frac{1}{4\rho_k}, \quad k = 1, \ldots, N \]

where \( \rho_N = 1 \) and

\[ \rho_k = \prod_{i=k+1}^{N} \xi_i + \lambda \sum_{i=k+1}^{N} x_i \prod_{j=k+1}^{i-1} \xi_j, \quad k = 1, \ldots, N - 1, \]

depends only on the fractions \( x_{k+1}, \ldots, x_N \).
Visualizing the resulting step sizes

Goal:

$$\min_{x_1, \ldots, x_N \in [0,1]} \mathbb{E} \left[ e^{H_N} + \lambda D_N \right]$$
Figure 2: Performance of proposed FH algorithm for spatial sampling for fixed $N$ samples. (a) Average entropy of hypothesis space after last sample. (b) Average distance traveled. (c) Average improvement of FH search from QS and UTB algorithms.
FH search summary

- Solve two-dimensional boundary detection problem as series of one-dimensional searches
- Optimal search policy in closed form
- Subsumes previous approaches as a special case ($N = 1$), significantly improves performance
- Doesn't solve the initial problem of interest
**Start:** Hypothesis/version space has length $|\mathcal{H}_0| = 1$
**Intuition: Active learning as a search**

**Start:** Hypothesis/version space has length $|\mathcal{H}_0| = 1$

\[0 \quad 1\]

**Goal:** Hypothesis space has length $|\mathcal{H}_N| = \varepsilon$

\[0.3 \quad 0.4\]
Stochastic Shortest Paths (SSP)
Stochastic Shortest Paths (SSP)

Diagram showing a grid with nodes A and B, and an arrow indicating a possible path.
Stochastic Shortest Paths (SSP)
Solve the SSP problem using dynamic programming (varying levels of computational efficiency)
**Figure 3:** Performance as a function of $T_t/T_s$. (a) Average number of samples. (b) Average distance traveled. (c) Improvement in average total sampling time between QS/UTB and proposed SPS algorithm.
Figure 4: Total sampling time as a function of change point $\theta$ for SPS algorithm with uniform prior (mean cost 130.56 sec) and non-uniform prior (mean cost 105.61 sec).
Figure 5: Number of charges as a function of change point $\theta$ for SPS algorithm with optimal recharging (mean 2.02 charges) vs naive recharging (mean 2.45 charges).
Thank you