engineering pipelines for learning at scale

Benjamin Recht
University of California, Berkeley
• If you can pose your problem as a simple optimization problem, *you’re mostly done*

• LASSO
• Support Vector Machines
• Logistic Regression
• Sparse approximation
• Graph cut problems
• Robust regression and classification
• Matrix completion
• Conditional random fields
• ...

\[
\begin{pmatrix}
\mathbf{M} \\
\boldsymbol{k \times n}
\end{pmatrix} =
\begin{pmatrix}
\mathbf{L} \\
\boldsymbol{k \times r}
\end{pmatrix}
\begin{pmatrix}
\mathbf{R}^* \\
\boldsymbol{r \times n}
\end{pmatrix}
\]

wavelet coefficients
large wavelet coefficients
wideband signal
large Gabor coefficients
time frequency

amazon.com
netflix.com
match.com
chemistry
SGD and Big Data at UW

(Stochastic Gradient Descent)

HOGWILD!
Run SGD in parallel without locks

JELLYFISH
Parallel SGD for Matrix Factorizations

BISMARCK
SGD inside the RDBMS
ML Pipelines

- Some problems just can’t be posed as an SVM.
- “Pipelined” engineering tools.

Challenges:

- (approximate) global optimization
- prototyping, simplification, and logical equivalence
- stability, convergence, safety
ML Design

Acquire → Normalize/Prune → Select Features → Classify

Text processing

Bag of Words → TFIDF → LDA → CRF

Object Recognition

Laplacian Pyramid → SIFT/HOG → Matching → k-NN
• Requires detailed hand charting by experts before data can be analyzed.
• Many details not recorded.
SportVU and the NBA

• Fancy, expensive 3D camera system
• 100K/year per team
• turning the stats into useful analytics is the challenge!
NFL Coaches Tape

- **NFL REWIND**: 5 seasons. 256 games per season. 120 plays per game. 2 views per play. Each play is 4 seconds on average plus pre- and post-roll (men in motion).
- 640x480 video, 30fps \(\approx 10\)TB of video.
- Low-quality, moving camera. Can we extract useful information?
OSU Digital Scout

Alan Fern, Rob Hess, Frank Liu
ECE Oregon State University

1. Register frames in panorama and snap panorama to field
2. Detect formation
3. Run a particle filter to track
1. Register frames in panorama and snap panorama to field

2. Detect formation

3. Run a particle filter to track
Pipeline for panorama:

1. find keypoints that are good for matching
2. create features at each keypoint and run nearest neighbors
3. compute homography to warp images
4. mix frames into panorama using graph cuts

Essentially Brown et al, CVPR 2005
Find Keypoints

- Shi and Tomasi
- many other options
- (12 options on wiki page!)

\[ R(p) = \lambda_{\text{min}} \left\{ \int \exp \left( -\gamma \|p - q\|^2 \right) \nabla I(q) \nabla I(q)^T dq \right\} \]

Make sure the keypoints are spread out over image

maximize_\Omega \sum_{p \in \Omega} R(p)
subject to \|p - q\| > \rho \quad \text{for all } p, q \in \Omega
|\Omega| = K
Match Features

- Image patches, SIFT, DAISY, DEEP BELIEF NETWORKS
- Feature engineering is huge

- Needs fast 2-nearest neighbor implementation.
Warp and cut

Search for best warp $W$ by robust least squares:

$$\min_W \sum_{p \in \mathcal{I}} (I_t(W(p)) - I_{t+1}(p) - z(p))^2 + \mu \sum_{p \in \mathcal{I}} |z(p)|$$

Pick one pixel from each image using graph cuts:

$$\min_x \sum_{(u,v) \in E} w_{uv} \|x_u - x_v\|_1$$
subject to $1^T_K x_v = 1$, $x_v \geq 0$, for $v = 1, \ldots, D$
- Run in OpenCV, C++, using beefy work station, not taking advantage of GPU or multicore

- Takes 10 hours. Best in breed?
- Could do more optimization, removing consecutive frames, other wizz bang trickery on GPU registers

**But how do we scale this process?**
1. Register frames in panorama and snap panorama to field

Pipeline for panorama:

1. find keypoints
2. prune keypoints
3. create features
4. run nearest neighbors
5. compute homography to warp images
6. mix frames into panorama using graph cuts

- convolution and local nonlinearity
- nonlinear optimization
- combinatorial optimization
• find the primitives
• optimize their implementation
• couple to mathematical optimization toolkit
Challenges

• **Optimized optimization primitives:** Build the necessary building blocks for discrete and continuous optimization in the cloud

• **Simplified pipeline synthesis:** Enable users to easily combine and prototype building blocks

• **Error analysis:** can we bound approximation errors or convergence rates for layered pipelines?
Optimize the primitives

- Basic BLAS + iterative solvers optimized for cloud infrastructure
- Can be called as part of an optimization subroutine.
- Enables training of 8TB model with 100m parameters in 2 hrs. on 200 machines.

$$\minimize \|Ax - b\|$$
Simplified synthesis

- Simple DSL for pipeline construction
- We can leverage functional programming concepts to chain nodes together.
- A simple DSL encodes the DAG, and is easy to extend.

Replicates Fast Food Features Pipeline - Le et. al., 2012

open source version coming soon…
Simplified analysis

- Use techniques from controls to analyze algorithms
- Automatically generate verification certificates
- Robustness analysis for complex, distributed systems

preprint at arxiv.org/abs/1408.3595
Challenges

- *(Approximate)* Global optimization: frame the whole problem as an optimization and solve it approximately

- Optimized pipelines: Figure out where redundant layers are or how reordering adds simplicity?

- Certification: can we automatically guarantee performance of complex pipelines?