Analyzing Task Driven Learning Algorithms

AMSC 663 Project Proposal

Mike Pekala

September 22, 2011

Advisor: Prof. Doron Levy (dlevy at math.umd.edu)
UMD Dept. of Mathematics & Center for Scientific Computation and Mathematical Modeling (CSCAMM)

Project Overview

The underlying notion of *sparse coding* is that, in many domains, data vectors can be concisely represented as a sparse linear combination of basis elements or dictionary atoms. Recent results suggest that, for many tasks, performance improvements can be realized by explicitly *learning* these dictionaries directly from the data (vs. using predefined dictionaries, such as wavelets) [5, 6]. Further results suggest that additional gains are possible by *jointly optimizing* the dictionary for both the data and the task (e.g. classification, denoising) [3].

For this project, we propose to

- Implement the task-driven dictionary algorithm defined in [3]
- Verify its correctness through unit testing and comparison with published performance results on standard machine learning data sets
- Analyze the performance on a new datset (e.g. hyperspectral imaging, vaccine data)

(Unless otherwise indicated, the notation/equations that follow are taken from [3])

Classical Dictionary Learning for Sparse Coding

Classical dictionary learning (e.g. [1]) does the following:

- Given: training data set of signals $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]$ in $\mathbb{R}^{m \times n}$
- Goal: design a dictionary $\mathbf{D}$ in $\mathbb{R}^{m \times p}$ (possible for $p > m$, i.e. an *overcomplete* dictionary) by minimizing the empirical cost function

$$g_n(\mathbf{D}) \triangleq \frac{1}{n} \sum_{i=1}^{n} \ell_u(\mathbf{x}_i, \mathbf{D})$$

where ℓ_u, the *unsupervised* loss function, is small when $\mathbf{D}$ is “good” at representing $\mathbf{x}_i$ sparsely. In [3], the authors use the elastic-net formulation:

$$\ell_u(\mathbf{x}, \mathbf{D}) \triangleq \min_{\boldsymbol{\alpha} \in \mathbb{R}^p} \frac{1}{2} ||\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}||_2^2 + \lambda_1 ||\boldsymbol{\alpha}||_1 + \frac{\lambda_2}{2} ||\boldsymbol{\alpha}||_2^2 \qquad (1)$$

Classical Dictionary Learning for Sparse Coding - Additional Details

- To prevent artificially improving ℓ_u by arbitrarily scaling $\mathbf{D}$, one typically constrains the set of permissible dictionaries:

$$\mathcal{D} \triangleq \{\mathbf{D} \in \mathbb{R}^{m \times p} \text{ s.t. } \forall j \in \{1, \ldots, p\}, ||\mathbf{d}_j||_2 \leq 1\}$$

- Optimizing the empirical cost g_n can be very expensive when the training set is large (as is often the case in dictionary learning problems). However, in reality, one usually wants to minimize the expected loss:

$$g(\mathbf{D}) \triangleq \mathbb{E}_{\mathbf{x}}\left[\ell_u(\mathbf{x}, \mathbf{D})\right] = \lim_{n \to \infty} g_n(\mathbf{D}) \quad \text{a.s.}$$

  (where expectation is taken with respect to the unknown distribution of data objects $p(\mathbf{x})$) In these cases, online, stochastic techniques have been shown to work well [4].

Task Driven Learning - Classification

Consider the classification task:

- Given: a learned dictionary $\mathbf{D}$ (fixed), an observation $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^m$ and a sparse representation of the observation $\mathbf{x} \approx \boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D})$ (e.g. determined using equation (1))
- Goal: identify the associated label $y \in \mathcal{Y}$, where $\mathcal{Y}$ is a finite set of labels (would be a subset of $\mathbb{R}^q$ for regression)

Assume $\mathbf{D}$ is fixed and $\boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D})$ will be used as the features for predicting y. The classification problem is to learn the model parameters $\mathbf{W}$ by solving:

$$\min_{\mathbf{W} \in \mathcal{W}} f(\mathbf{W}) + \frac{\nu}{2} ||\mathbf{W}||_F^2$$

where

$$f(\mathbf{W}) \triangleq \mathbb{E}_{y,\mathbf{x}} \left[\ell_s(y, \mathbf{W}, \boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D})) \right]$$

and ℓ_s is a convex loss function (e.g. logistic) adapted to the supervised learning problem.

Task Driven Learning - Classification

Task driven learning adds the notion of a *supervised task* to the unsupervised dictionary learning problem. To make a task driven classification problem, want to jointly learn $\mathbf{D}, \mathbf{W}$:

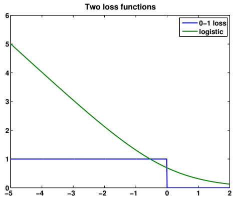
$$\min_{\mathbf{D}\in\mathcal{D},\mathbf{W}\in\mathcal{W}} f(\mathbf{D},\mathbf{W}) + \frac{\nu}{2}||\mathbf{W}||_F^2 \tag{2}$$

where

$$f(\mathbf{D},\mathbf{W}) \triangleq \mathbb{E}_{y,\mathbf{x}}\left[\ell_s(y,\mathbf{W},\boldsymbol{\alpha}^\star(\mathbf{x},\mathbf{D}))\right]$$

Example:

Binary classification: $\mathcal{Y} = \{-1,+1\}$
Linear model: $\mathbf{w} \in \mathbb{R}^p$
Prediction: $m = \mathbf{w}^T\boldsymbol{\alpha}^\star(\mathbf{x},\mathbf{D})$
Logistic loss: $\ell_s = \log\left(1+e^{-ym}\right)$



$$\min_{\mathbf{D}\in\mathcal{D},\mathbf{w}\in\mathbb{R}^p} \mathbb{E}_{y,\mathbf{x}}\left[\log\left(1+e^{-y\mathbf{w}^T\boldsymbol{\alpha}^\star(\mathbf{x},\mathbf{D})}\right)\right] + \frac{\nu}{2}||\mathbf{w}||_2^2 \tag{3}$$

Solving the Problem

Stochastic gradient descent is often used to minimize functions whose gradients are expectations. The authors of [3] show that, under suitable conditions, equation (2) is differentiable on $\mathcal{D} \times \mathcal{W}$, and that

$$\begin{aligned} \nabla_{\mathbf{W}} f(\mathbf{D}, \mathbf{W}) &= \mathbb{E}_{y,\mathbf{x}} \left[\nabla_{\mathbf{W}} \ell_s(y, \mathbf{w}, \boldsymbol{\alpha}^\star) \right] \\ \nabla_{\mathbf{D}} f(\mathbf{D}, \mathbf{W}) &= \mathbb{E}_{y,\mathbf{x}} \left[-\mathbf{D}\boldsymbol{\beta}^\star \boldsymbol{\alpha}^{\star T} + (\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}^\star)\boldsymbol{\beta}^{\star T} \right] \end{aligned}$$

where $\boldsymbol{\beta}^\star \in \mathbb{R}^p$ is defined by the properties:

$$\boldsymbol{\beta}^\star_{\Lambda^C} = 0 \text{ and } \boldsymbol{\beta}^\star_\Lambda = (D_\Lambda^T D_\Lambda + \lambda_2 \mathbf{I})^{-1} \nabla_{\alpha_\Lambda} \ell_s(y, \mathbf{W}, \boldsymbol{\alpha}^\star)$$

and Λ are the indices of the nonzero coefficients of $\boldsymbol{\alpha}^\star(\mathbf{x}, \mathbf{D})$.

Algorithm: SGD for task-driven dictionary learning [3]

Input: $p(y, \mathbf{x})$ (a way to draw samples i.i.d. from p), $\lambda_1, \lambda_2, \nu \in \mathbb{R}$ (regularization parameters), $\mathbf{D} \in \mathcal{D}$ (initial dictionary), $\mathbf{W} \in \mathcal{W}$ (initial model), T (num. iterations), $t_0, \rho \in \mathbb{R}$ (learning rate parameters)

```
1 for $t = 1$ to $T$ do
2     Draw $(y_t, \mathbf{x}_t)$ from $p(y, \mathbf{x})$
3     Compute $\boldsymbol{\alpha}^\star$ via sparse coding
4     Compute active set $\Lambda$
5     Compute $\boldsymbol{\beta}^\star$
6     Update learning rate $\rho_t$
7     Update parameters
8 end
```

Algorithm: SGD for task-driven dictionary learning [3]

Input: $p(y, \mathbf{x})$ (a way to draw samples i.i.d. from p), $\lambda_1, \lambda_2, \nu \in \mathbb{R}$ (regularization parameters), $\mathbf{D} \in \mathcal{D}$ (initial dictionary), $\mathbf{W} \in \mathcal{W}$ (initial model), T (num. iterations), $t_0, \rho \in \mathbb{R}$ (learning rate parameters)

```
1 for $t = 1$ to $T$ do
2     Draw $(y_t, \mathbf{x}_t)$ from $p(y, \mathbf{x})$ ◀
3     Compute $\boldsymbol{\alpha}^\star$ via sparse coding
4     Compute active set $\Lambda$
5     Compute $\boldsymbol{\beta}^\star$
6     Update learning rate $\rho_t$
7     Update parameters
8 end
```

Obtaining i.i.d. samples may be difficult since $p(y, \mathbf{x})$ is unknown. As an approximation, vectors $(y_t, \mathbf{x}_t)$ obtained by cycling over a randomly permuted training set.

Algorithm: SGD for task-driven dictionary learning [3]

Input: $p(y, \mathbf{x})$ (a way to draw samples i.i.d. from p), $\lambda_1, \lambda_2, \nu \in \mathbb{R}$ (regularization parameters), $\mathbf{D} \in \mathcal{D}$ (initial dictionary), $\mathbf{W} \in \mathcal{W}$ (initial model), T (num. iterations), $t_0, \rho \in \mathbb{R}$ (learning rate parameters)

```
1 for $t = 1$ to $T$ do
2     Draw $(y_t, \mathbf{x}_t)$ from $p(y, \mathbf{x})$
3     Compute $\boldsymbol{\alpha}^\star$ via sparse coding ◀
4     Compute active set $\Lambda$
5     Compute $\boldsymbol{\beta}^\star$
6     Update learning rate $\rho_t$
7     Update parameters
8 end
```

Use a modified LARS [2] algorithm (or equivalent) to solve:

$$\boldsymbol{\alpha}^\star = \arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^p} \frac{1}{2}||\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}||_2^2 + \lambda_1 ||\boldsymbol{\alpha}||_1 + \frac{\lambda_2}{2}||\boldsymbol{\alpha}||_2^2$$

This is a nontrivial sub-project.

Algorithm: SGD for task-driven dictionary learning [3]

Input: $p(y, \mathbf{x})$ (a way to draw samples i.i.d. from p), $\lambda_1, \lambda_2, \nu \in \mathbb{R}$ (regularization parameters), $\mathbf{D} \in \mathcal{D}$ (initial dictionary), $\mathbf{W} \in \mathcal{W}$ (initial model), T (num. iterations), $t_0, \rho \in \mathbb{R}$ (learning rate parameters)

```
1 for $t = 1$ to $T$ do
2     Draw $(y_t, \mathbf{x}_t)$ from $p(y, \mathbf{x})$
3     Compute $\boldsymbol{\alpha}^\star$ via sparse coding
4     Compute active set $\Lambda$ ◀
5     Compute $\boldsymbol{\beta}^\star$ ◀
6     Update learning rate $\rho_t$
7     Update parameters
8 end
```

Λ are indices of non-zeros in $\boldsymbol{\alpha}^\star$

$$\boldsymbol{\beta}^\star_{\Lambda^C} = 0$$

$$\boldsymbol{\beta}^\star_{\Lambda} = (D_\Lambda^T D_\Lambda + \lambda_2 \mathbf{I})^{-1} \nabla_{\alpha_\Lambda} \ell_s(y, \mathbf{W}, \boldsymbol{\alpha}^\star)$$

Algorithm: SGD for task-driven dictionary learning [3]

Input: $p(y, \mathbf{x})$ (a way to draw samples i.i.d. from p), $\lambda_1, \lambda_2, \nu \in \mathbb{R}$ (regularization parameters), $\mathbf{D} \in \mathcal{D}$ (initial dictionary), $\mathbf{W} \in \mathcal{W}$ (initial model), T (num. iterations), $t_0, \rho \in \mathbb{R}$ (learning rate parameters)

```
1 for $t = 1$ to $T$ do
2     Draw $(y_t, \mathbf{x}_t)$ from $p(y, \mathbf{x})$
3     Compute $\boldsymbol{\alpha}^\star$ via sparse coding
4     Compute active set $\Lambda$
5     Compute $\boldsymbol{\beta}^\star$
6     Update learning rate $\rho_t$ ◀
7     Update parameters
8 end
```

$$\rho_t = \min(\rho, \tfrac{t_0}{t}\rho)$$

There is some art in selecting learning rates for faster convergence (possible complexity issue).

Algorithm: SGD for task-driven dictionary learning [3]

Input: $p(y, \mathbf{x})$ (a way to draw samples i.i.d. from p), $\lambda_1, \lambda_2, \nu \in \mathbb{R}$ (regularization parameters), $\mathbf{D} \in \mathcal{D}$ (initial dictionary), $\mathbf{W} \in \mathcal{W}$ (initial model), T (num. iterations), $t_0, \rho \in \mathbb{R}$ (learning rate parameters)

```
1 for $t = 1$ to $T$ do
2     Draw $(y_t, \mathbf{x}_t)$ from $p(y, \mathbf{x})$
3     Compute $\boldsymbol{\alpha}^\star$ via sparse coding
4     Compute active set $\Lambda$
5     Compute $\boldsymbol{\beta}^\star$
6     Update learning rate $\rho_t$
7     Update parameters ◀
8 end
```

$$\mathbf{W} = \Pi_{\mathcal{W}} \left[\mathbf{W} - \rho_t \left(\nabla_{\mathbf{W}} \ell_s(y, \mathbf{w}, \boldsymbol{\alpha}^\star)\right) + \nu \mathbf{W}\right]$$

$$\mathbf{D} = \Pi_{\mathcal{D}} \left[\mathbf{D} - \rho_t \left(-\mathbf{D}\boldsymbol{\beta}^\star \boldsymbol{\alpha}^{\star T} + (\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}^\star)\boldsymbol{\beta}^{\star T}\right)\right]$$

where $\Pi_{\mathcal{S}}$ denotes orthogonal projection on the set $\mathcal{S}$. Difficulty depends upon the constraints associated with $\mathcal{S}$.

Learning Linear Transforms of Input Data

The authors of [3] also provide additional extensions to this framework. For example, jointly learning a linear transform Z of the input data,

$$\min_{\mathbf{D}\in\mathcal{D},\mathbf{W}\in\mathcal{W},\mathbf{Z}\in\mathcal{Z}} f(\mathbf{D},\mathbf{W},\mathbf{Z}) + \frac{\nu_1}{2}||\mathbf{W}||_F^2 + \frac{\nu_2}{2}||\mathbf{Z}||_F^2$$

$$f(\mathbf{D},\mathbf{W},\mathbf{Z}) \triangleq \mathbb{E}_{y,\mathbf{x}}\left[\ell_s(y,\mathbf{W},\boldsymbol{\alpha}^\star(\mathbf{Zx},\mathbf{D})\right]$$

These extensions may **optionally** be considered in this project, depending on initial progress/results.

Project Details

Software Implementation

- Platform: OSX and Linux, Matlab/Octave, gcc + any standard packages
- Implementation Language: Matlab, possibly with extensions in C/C++ via mex, depending on performance/profiler results. Reserve the right to use Python or other standard tools for data pre/post processing.
- Complexity Issues: Alluded to possible issues with SGD, projection operations. Performance of LARS also unclear at this point. Dictionary size a possible concern.

Deliverables

- Deliverables for this project include all source code and a written report of results obtained on all data sets. The databases themselves are **not** a deliverable for this project.

Validation Datasets

LARS

- For the LARS algorithm [2], the authors provide a data set which contains 10 baseline variables (e.g. age, sex, BMI) for 442 diabetes patients along with a quantitative measure of disease progress after one year. We will validate our implementation of LARS by comparing results obtained with our implementation (when $\lambda_2 = 0$). For $\lambda_2 > 0$, we will validate against hand-constructed problems with known solutions.

Task-driven dictionary learning

- The authors of [3] provide performance results for a classification task using two standard machine learning datasets, MNIST and USPS (handwriting recognition problems). We will validate our implementation of LARS+TDDL by replicating their experimental setup.

Analysis Datasets

The analysis portion of this project involves datasets for which there are not known results for this particular algorithm. Final choice of analysis dataset is TBD (will be included in proposal); potential candidates include:

- Hyperspectral Imaging Data
  - Spectral range: 400-1000nm (visible and near-infrared)
  - Spectral resolution: $\mathcal{O}(10\text{nm})$
  - Spatial resolution: $\mathcal{O}(\text{meters})$
  - Image size: $\sim 1024 \times 1024$
- Vaccine Data

Schedule and Milestones

Schedule and milestones will be updated/maintained in the project proposal (living document).

- Phase I: Algorithm development (Sept 23 - Jan 15)
  - Phase Ia: Implement LARS (Sept 23 $\sim$ Oct 24)
    - Milestone: LARS code available
  - Phase Ib: Validate LARS (Oct 24 $\sim$ Nov 14)
    - Milestone: results on diabetes dataset and hand-crafted problems
  - Phase Ic: Implement SGD framework (Nov 14 $\sim$ Dec 15)
    - Milestone: Initial SGD code available
  - Phase Id: Validate SGD framework (Dec 15 $\sim$ Jan 15)
    - Milestone: TDDL results on MNIST and USPS.
- Phase II: Analysis on new data sets (Jan 15 - May 1)
  - Milestone: Preliminary results on selected dataset ($\sim$ Mar 1)
  - Milestone: Final report and presentation ($\sim$ May 1)

Potential Issue: Size of Dictionary

- For large data sets, may not be able to fit entire dictionary into memory
- Can reduce size of data (e.g. PCA), partition into patches, etc.
- Another option: memory mapping (for "lazy loading")

Example

In OMP, the "proxy calculation" step is potentially memory-prohibitive:

$$p_t = \arg \max_{j=1,\dots,p} |\langle \mathbf{r}_{t-1}, \mathbf{d}_j \rangle|$$

Instead of loading **D** into memory, incrementally load subsets. Not real fast (at least as I implemented it); parallelism also reasonable here.

Memory Mapping Example: Create MMap

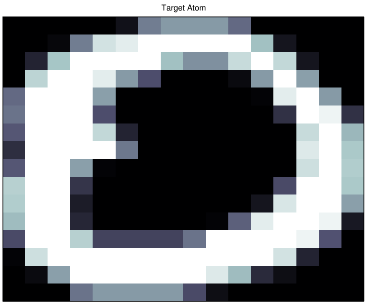
```
% Creates a memory mapped dictionary from the training portion of
% the USPS data set.

X = load_usps();

% create a big dictionary (> 8GB)
[BLOCK_SIZE, N_BLOCKS] = usps_mm_params();
fd = fopen('uspsdict.raw', 'wb');
for ii = 1:N_BLOCKS
  if (mod(ii,100) == 1)  fprintf(1,'[info]: writing block %d (%%%.0f)\n', ii, floor(100*ii/N_BLOCKS)); en

  % choose a subset of the data set
  indices = randperm(size(X,1)); indices = indices(1:BLOCK_SIZE);

  % convert row data to columns and add some noise
  Xii = X(indices,:)';
  Xii = Xii + 0.05*randn(size(Xii));

  % write the block
  fwrite(fd, Xii, 'double');
end
```

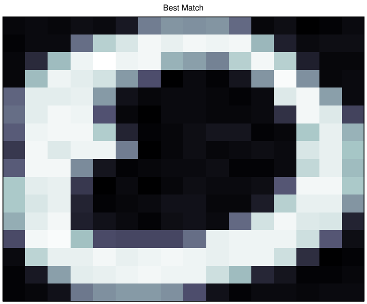
Memory Mapping Example: Use MMap

```
% run the proxy selection stage of OMP (basically a matched filter)
% against some USPS examples.
clear all; close all;

[BLOCK_SIZE, N_BLOCKS] = usps_mm_params();
mm = memmapfile('uspsdict.raw', 'Format', ...
                {'double', [16*16, BLOCK_SIZE], 'atoms'});

X = load_usps();
X = X';  % convert row data to columns

tgtAtomIdx = floor(rand(1)*size(X,2))+1;
tgtAtom = X(:,tgtAtomIdx); % look for an atom similar to this one

ipMax = 0; blockIdMax = -1; atomIdMax = -1;
tic
for blockIdx = 1:N_BLOCKS  % find the best matching atom
  if (mod(blockIdx,100) == 1)  fprintf(1,'[info]: searching block %d (%%%.0f)\n', blockIdx, floor(100*blo

  block = mm.Data(blockIdx).atoms;          % load block into memory
  [val,atomIdx] = max(abs(block'*tgtAtom)); % calculate proxy
  if val > ipMax
    ipMax = val;
    blockIdMax = blockIdx;
    atomIdMax = atomIdx;
  end
end

matchAtom = mm.Data(blockIdMax).atoms(:,atomIdMax);
toc

% plot results
plot_atom_usps(tgtAtom);  title('Target Atom');  saveas(gcf, 'target.eps', 'epsc');
plot_atom_usps(matchAtom);  title('Best Match'); saveas(gcf, 'bestmatch.eps', 'epsc');
```

Memory Mapping Example: Example

```
>> mf_mmap_usps
[info]: searching block 1 (%0)
[info]: searching block 101 (%2)
...
[info]: searching block 4901 (%98)
Elapsed time is 319.995970 seconds.
```





Bibliography I

[1] Michal Aharon, Michael Elad, and Alfred Bruckstein. K-SVD: Design of dictionaries for sparse representation. In *IN: PROCEEDINGS OF SPARS05*, pages 9–12, 2005.

[2] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

[3] Julien Mairal, Francis Bach, and Jean Ponce. Task-Driven Dictionary Learning. Rapport de recherche RR-7400, INRIA, 2010.

[4] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 689–696, New York, NY, USA, 2009. ACM.

[5] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the Twenty-fourth International Conference on Machine Learning*, 2007.

Bibliography II

[6] Ignacio Ramrez, Pablo Sprechmann, and Guillermo Sapiro. Classification and clustering via dictionary learning with structured incoherence and shared features. In *CVPR*, pages 3501–3508. IEEE, 2010.