Task-driven dictionary learning

Presented by Brian McFee for CSE254, W2012

Julien Mairal, Francis Bach, and Jean Ponce
A typical machine learning pipeline

1. Collect data
2. Extract features
3. Train predictor
A deep learning pipeline

- **Input layer**
- **Feature layers** (unsupervised)
- **Output layer** (supervised)
Background: sparse coding

Given data $X \in \mathbb{R}^{d \times n}$

$$X = \begin{bmatrix} x^1 & x^2 & \cdots & x^n \end{bmatrix}$$

Learn a **dictionary** $D \in \mathbb{R}^{d \times k}$ **encoding** $A \in \mathbb{R}^{k \times n}$ ($k > d$)

$$X \approx DA$$

**Sparse** $A = $ each $x^i$ explained by few codewords
Sparse coding vs PCA

PCA output

(input data)

(dense, low-dimensional)
Sparse coding vs PCA

**PCA output**

(input data)

(dense, low-dimensional)

**Sparse coding output**

(input data)

(sparse, high-dimensional)
Why code sparsely?

Biologically:
Neurons don't all fire at once
(conserve energy)

Practically:
Works well for classification
How to code sparsely

Given $x$ and $D$, define energy function

$$E(a; x, D) := \frac{1}{2} \|x - Da\|^2 + \lambda \|a\|_1$$

- **Accuracy**
- **Sparsity**
How to code sparsely

Given $x$ and $D$, define energy function

$$E(a; x, D) := \frac{1}{2}\|x - Da\|^2 + \lambda\|a\|_1$$
Learning the dictionary

Given $n$ samples $x^i \in \mathbb{R}^d$, minimize total energy:

$$\min_{D,A} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} \| x^i - Da^i \|^2 + \lambda \| a^i \|_1$$

s.t. $\forall j : \| d^j \|^2 \leq 1$

Problem: not jointly convex in $D,A$
Learning by alternating minimization

But it is convex in \(D\) and \(A\) individually

Fix \(D\), optimize \(A\)

(Lasso * \(n\))
Learning by alternating minimization

But it is convex in $D$ and $A$ individually

Fix $D$, optimize $A$ (Lasso * $n$)

Fix $A$, optimize $D$ (QCQP)
Learning by alternating minimization

But it is convex in $D$ and $A$ individually

Fix $D$, optimize $A$

(Lasso $\times n$)

Fix $A$, optimize $D$

(QCQP)

2000 YEARS LATER
Supervised learning

Idea: use encodings $a$ as input to a predictor

Ex: linear regression, $y \in \mathbb{R}^m$, $W \in \mathbb{R}^{m \times k}$

$$E(y; W, a) := \frac{1}{2} \|y - Wa\|^2$$
A disconnected architecture

The usual plan:

1. Learn dictionary $D$
2. Learn predictor $W$
3. Fame and glory

But these are tasks not independent!
Unified architecture

Jointly learn codebook $D$ and predictor $W$
Joint optimization: round 1

Given $n$ samples $(x^i, y^i)$, minimize energy:

$$\min_{D,A,W} \left( \sum_{i=1}^{n} E(y^i; W, a^i) + \lambda_0 E(a^i; x^i, D) \right) + \frac{\nu}{2} \|W\|_F^2$$
Alternating minimization

Fix \((W, D)\), update \(A\)  
Fix \(A\), update \((W,D)\)
At prediction time...

For fixed $W$, $D$ and test input $x$, compute

$$\arg \min_{y, a} E(y; W, a) + \lambda_0 E(a; x, D)$$

Easy for linear regression:

$$E(y; W, a) := \frac{1}{2} \| y - Wa \|^2$$

$$y = Wa \Rightarrow E(y; W, a) = 0$$
At prediction time...

What about categorical output? (e.g. $y \in \{0,1\}$)

Encoding $a$ depends on input AND output

Seems unnatural, complicates prediction
Code first, predict later

Old learning problem (sum of energies):

$$\min_{D,A,W} \left( \sum_{i=1}^{n} E(y^i; W, a^i) + \lambda_0 E(a^i; x^i, D) \right) + \frac{\nu}{2} \| W \|_F^2$$

[Mairal, Bach and Ponce, 2011]
Old learning problem (sum of energies):

$$\min_{D,A,W} \left( \sum_{i=1}^{n} E(y^i; W, a^i) + \lambda_0 E(a^i; x^i, D) \right) + \frac{\nu}{2} \|W\|_F^2$$

New formulation (optimal encoding):

$$\min_{D,W} \sum_{i=1}^{n} E(y^i; W, a^*(x^i, D)) + \frac{\nu}{2} \|W\|_F^2$$

$$a^*(x, D) = \arg\min_a E(a; x, D)$$
Optimal encoding

Output is now a function of optimal encoding

\[
\min_{D, W} \sum_{i=1}^{n} E \left( y^i; W, a^*(x^i, D) \right) + \frac{\nu}{2} \| W \|_F^2
\]

Prediction is feed-forward:

a. Encode \( x \) as \( a^* \)

b. Predict \( y \)

c. Done.

But how to learn parameters \( W, D \)?
Stochastic gradient descent

Minimize objective in expectation:

\[
\min_{D,W} f(D, W) \\
\quad f(D, W) := \mathbb{E}_{(x,y)} [E(y; W, a^*(x, D))] + \frac{\nu}{2} \|W\|_F^2
\]
Stochastic gradient descent

Minimize objective in expectation:

$$\min_{D,W} f(D, W)$$

$$f(D, W) := \mathbb{E}_{(x,y)} [E(y; W, a^*(x, D))] + \frac{\nu}{2}\|W\|_F^2$$

**Algorithm**: repeat

a. Pick a random \((x,y)\) from training data
b. Update \(D, W\)
**Stochastic gradient descent**

Minimize objective in expectation:

\[
\min_{D,W} f(D, W)
\]

\[
f(D, W) := \mathbb{E}_{(x,y)}[E(y; W, a^*(x, D))] + \frac{\nu}{2} \|W\|_F^2
\]

**Algorithm**: repeat

a. Pick a random \((x, y)\) from training data
b. Update \(D, W\)

Need two quantities:

\[
\nabla_W f(D, W) \\
\nabla_D f(D, W)
\]
W update

Easy for differentiable $E$:

$$W \leftarrow W - \eta \nabla_W \left[ E(y; W, a^*) + \frac{v}{2} \| W \|_F^2 \right]$$

for learning rate $\eta$
D update?

\( a^*(x, D) \) is not differentiable

\[
a^*(x, D) = \arg\min_a \frac{1}{2} \| x - Da \|^2 + \lambda \| a \|_1
\]

\[
\nabla_D a^*(x, D) = ???
\]
Key observation

Small change in $D$

Small change in $a^*(x, D)$

Differentiable region

$a^*(x, D') \approx a^*(x, D)$
Add a compression layer: $a^*(Zx, D)$

$y$: output
$a$: coding
$z$: compression
$x$: input

Deeper?
Experiments
Exp. 1: Hand-written digit recognition

1-vs-all logistic regression (10 classes)

MNIST: 60K train, 10K test, 28x28
USPS:  7.3K train, 2K test, 16x16

<table>
<thead>
<tr>
<th>$D$</th>
<th>Unsupervised</th>
<th>Supervised</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k^*$</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>MNIST</td>
<td>5.27</td>
<td>3.92</td>
</tr>
<tr>
<td>USPS</td>
<td>8.02</td>
<td>6.03</td>
</tr>
</tbody>
</table>

% Error
Exp. 2: Inverse half-tone

Predict grayscale image from black&white input

Linear regression

10x10 image patches

k=500 codewords

36 images, 9M patches

Original (8bpp)  | Observed (1bpp)  | Reconstructed (8bpp)
Exp. 2: Inverse half-tone results
Conclusion

Supervision can help learn good features