(Unsupervised) Feature Learning +
(Supervised) Machine Learning
(Self Taught) Deep Learning

Lecture 05

Razvan C. Bunescu
School of Electrical Engineering and Computer Science
bunescu@ohio.edu
Sparse Coding

• A **coder** learns a possibly **over-complete** set of **basis vectors** $\Phi_k$ and **features** $a_k$ such that an input vector $x$ can be represented as a linear combination:

$$x = \sum_{k=1}^{K} a_k \Phi_k$$

• **Problem**: over-complete basis $\Rightarrow$ given $\Phi_k$, the features $a_k$ are no longer uniquely determined by $x$.

• **Solution**: require features to be sparse.
  1) Few $a_k$ are non-zero.
  2) Few $a_k$ are not close to zero.
Sparse Coding

• A **sparse coder** learns a possibly **over-complete** set of **basis vectors** $\Phi_k$ and **sparse features** $a_k$ such that:

$$\hat{x} = \sum_{k=1}^{K} a_k \Phi_k$$

• For a dataset $X = \{x^{(j)}\}_{1 \leq j \leq m}$, the cost function becomes the **reconstruction error** + **sparsity term**:

$$J(a, \Phi) = \sum_{j=1}^{m} \left( \left\| x^{(j)} - \sum_{k=1}^{K} a_k^{(j)} \Phi_k \right\|_2^2 + \lambda \sum_{k=1}^{K} s(a_k^{(j)}) \right)$$

• What sparsity function $s(a)$ to use here?
Sparse Coding

- Cost function is the reconstruction error + sparsity term:

\[
J(a, \Phi) = \sum_{j=1}^{m} \left( \left\| x^{(j)} - \sum_{k=1}^{K} a_k^{(j)} \Phi_k \right\|^2_2 + \lambda \sum_{k=1}^{K} s(a_k^{(j)}) \right)
\]

- Sparsity functions:
  1) Few \( a_k \) are non-zero:

\[
s(a_k) = 1(|a_k| > 0)
\]

  2) Few \( a_k \) are not close to zero:

\[
s(a_k) = \frac{1}{\log(1 + a_k^2)}
\]

\[
s(a_k) = \sqrt{a_k^2 + \varepsilon}
\]
Sparse Coding: Final Touch

• Cost function is the **reconstruction error** + **sparsity term**:

\[
J(a, \Phi) = \sum_{j=1}^{m} \left( \left\| x^{(j)} - \sum_{k=1}^{K} a_k^{(j)} \Phi_k \right\|^2 + \lambda \sum_{k=1}^{K} s\left(a_k^{(j)}\right) \right)
\]

• But sparsity term can be arbitrarily small by proper rescaling of **basis vectors** $\Phi_k$ and **sparse features** $a_k$.
  – can scale down $a_k$ and scale up $\Phi_k$ by some large constant.

• Constrain $\Phi_k$ to have a bounded $L_2$ norm.

\[
\text{minimize: } J(a, \Phi) = \sum_{j=1}^{m} \left( \left\| x^{(j)} - \sum_{k=1}^{K} a_k^{(j)} \Phi_k \right\|^2 + \lambda \sum_{k=1}^{K} s\left(a_k^{(j)}\right) \right)
\]

subject to: $\|\Phi_k\|_2^2 \leq 1$
Sparse Coding: Final Touch

- The constraint $\|\Phi_k\|_2^2 \leq 1$ cannot be enforced using simple gradient descent methods.

- Replace constraint with a weight decay term:

\[
J(a, \Phi) = \sum_{j=1}^{m} \left( \| \mathbf{x}^{(j)} - \sum_{k=1}^{K} a_k^{(j)} \Phi_k \|_2^2 + \lambda \sum_{k=1}^{K} s\left(a_k^{(j)}\right) + \gamma \sum_{k=1}^{K} \| \Phi_k \|_2^2 \right)
\]

\[
s(a_k) = \sqrt{a_k^2 + \varepsilon}
\]
Sparse Autoencoder vs. Sparse Coding

- A **sparse auto-encoder** learns a set of weights $W^{(1)}$ that give us a set of sparse features $a$ that can reconstruct $x$:

  $$a = \sigma(W^{(1)}x + b^{(1)})$$

  $$\hat{x} = W^{(2)}a + b^{(2)} \Rightarrow \hat{x}_i = \sum_{k=1}^{K} W_{ik}^{(2)} a_k + b_i^{(2)} \Rightarrow \hat{x} = \sum_{k=1}^{K} W_{:k}^{(2)} a_k + b^{(2)}$$

  $$\hat{x} = \sum_{k=1}^{K} a_k \Phi_k + b^{(2)}$$

- A **sparse coder** learns the set of sparse features $a$ directly, together with a set of basis vectors $\Phi$:

  $$\hat{x} = \sum_{k=1}^{K} a_k \Phi_k$$
PCA vs. Sparse Coding

- **PCA** finds an **under-complete** set of basis vectors $U_{1,K}$ such that:
  \[
  \hat{y} = U_{1,K}^T x = [u_1, \ldots, u_K]^T x \Rightarrow \hat{y}_k = u_k^T x
  \]
  \[
  \hat{x} = U_{1,K} \hat{y} = \sum_{k=1}^{K} \hat{y}_k u_k \Rightarrow \hat{x} = \sum_{k=1}^{K} a_k \Phi_k
  \]

- A **sparse coder** can learn an **over-complete** set of basis vectors $\Phi$:
  \[
  \hat{x} = \sum_{k=1}^{K} a_k \Phi_k
  \]
Sparse Coding: Vectorization

- The non-vectorized cost function is:

\[
J(\mathbf{a}, \Phi) = \sum_{j=1}^{m} \left( \left\| \mathbf{x}^{(j)} - \sum_{k=1}^{K} a^{(j)}_{k} \Phi_{k} \right\|_{2}^2 + \lambda \sum_{k=1}^{K} s\left( a^{(j)}_{k} \right) \right) + \gamma \sum_{k=1}^{K} \|\Phi_{k}\|_{2}^2
\]

where \( s(a_{k}) = \sqrt{a_{k}^2 + \varepsilon} \)

- Let \( \Phi = [\Phi_1, ..., \Phi_K] \) and \( A = [\mathbf{a}^{(1)}, ..., \mathbf{a}^{(m)}] \).

- The equivalent vectorized cost function is:

\[
J(A, \Phi) = \|\Phi A - X\|_{2}^2 + \lambda S(A) + \gamma \|\Phi\|_{2}^2
\]
Sparse Coding: Optimization Algorithm

I. Initialize $\Phi$ randomly.

II. Repeat until convergence:
   1. Find $A$ that minimizes $J(A, \Phi)$ for the current $\Phi$.
   2. Find $\Phi$ that minimizes $J(A, \Phi)$ for the current $A$.

• Batching examples into mini-batches:
  – At each iteration, selected a random subset (2,000) from the whole dataset (10,000).

• Good initialization of $A$:
  – Set $A = \Phi^TX$, where $X$ is the random mini-batch.
  – Divide each column in $A$ by the norm of the corresp. column in $\Phi$.

Lecture 5
Sparse Coding: Optimization Algorithm

I. Initialize $\Phi$ randomly.

II. Repeat until convergence:
   1. Select a random mini-batch (2,000 patches) into $X$.
   2. Initialize $A$:
      - Let $A = \Phi^T X$
      - For each column $A_c$ of $A$:
        - Let $A_c = A_c / ||\Phi_c||$
   3. Find $A$ that minimizes $J(A, \Phi)$ for the current $\Phi$.
   4. Find $\Phi$ that minimizes $J(A, \Phi)$ for the current $A$. 

Lecture 5
Sparse Coding: Gradients

- Vectorized cost function:

\[ J(A, \Phi) = \| \Phi A - X \|_2^2 + \lambda S(A) + \gamma \| \Phi \|_2^2 \]

- Vectorized gradients:

\[ \nabla_{\Phi} J(A, \Phi) = 2(\Phi A - X) A^T + 2\gamma I \]

\[ \nabla_{\theta} J(A, \Phi) = 0 \Rightarrow \Phi^* = X A^T \left( A A^T + \gamma I \right)^{-1} \]

\[ \nabla_A J(A, \Phi) = 2\Phi^T (\Phi A - X) + \lambda T(A) \]

\[ t(a_{ij}) = \frac{a_{ij}}{\sqrt{a_{ij}^2 + \varepsilon}} \]
Features from Normal Sparse Coding
Topographic Sparse Coding

- Want adjacent features in A to be similar:
  - For each example $x$, organize features $a$ into a (square) matrix.
  - Want adjacent features in the matrix $a$ to have similar values.
  - Group adjacent features in the smoothed $L_1$ penalty:
    - Replace $\sqrt{a_{11}^2 + \varepsilon}$ with $\sqrt{a_{11}^2 + a_{12}^2 + a_{13}^2 + a_{21}^2 + a_{22}^2 + a_{23}^2 + a_{31}^2 + a_{32}^2 + a_{33}^2 + \varepsilon}$
    - Do this for every group (3x3 region) in the feature matrix:
      - Wrap around (matrix as a torus) $\Rightarrow mK$ groups in total.

- Topographic cost function:
  $$J(A, \Phi) = \|\Phi A - X\|_2^2 + \gamma \|\Phi\|_2^2 + \lambda \sum_{j=1}^{m} \left( \sum_{g \in \text{Groups}(a^{(j)})} \sqrt{\sum_{a_{rc} \in g} a_{rc}^2} + \varepsilon \right)$$
Features from Topographic Sparse Coding
Sparse Coding vs. Sparse AutoEncoder

- **sCoder**, training time:
  - Learn set $\Phi$ of basis vectors.

- **sCoder**, test time:
  - Need to encode new example $x$:
    - Find $A$ that minimizes $J(A, \Phi)$ for the current $\Phi$.
    - Run optimization, which can be time consuming.

- **sAutoEncoder**, training time:
  - Learn encoding parameters $W^{(1)}$.

- **sAutoEncoder**, test time:
  - Need to encode new example $x$:
    - Run one step of feedforward propagation, which is very fast.