COMP 790-125: Goals for today

- Matching pursuit
- Learning dictionaries
  - batch learning
  - online learning
- Sparse representations, compressive sensing

Dictionary based representations

Last time we talked about sparsely representing a patch

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \frac{1}{2} \left\| \underbrace{\mathbf{R}_{i,j}\mathbf{y}}_{\text{extracted patch}} - \underbrace{\mathbf{D}\boldsymbol{\alpha}}_{\substack{\text{combination of} \\ \text{dictionary elements}}} \right\|_2^2 + \lambda \underbrace{\|\boldsymbol{\alpha}\|_1}_{\text{sparsity penalty}}$$

We talked about using dictionaries to obtain denoised images

$$\underset{\boldsymbol{\alpha},\mathbf{x}}{\operatorname{argmin}} \frac{\mu}{2} \|\mathbf{x}-\mathbf{y}\|_2^2 + \sum_{i,j} \left\{ \frac{1}{2} \|\mathbf{R}_{i,j}\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}_{i,j}\|_2^2 + \lambda \|\boldsymbol{\alpha}_{i,j}\|_1 \right\}$$

All the while, we assumed that we know $\mathbf{D}$, and we looked at some standard (orthonormal) dictionaries.

Matching pursuit

One of the benefits of having an orthogonal dictionary $\mathbf{D}$ is that

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{y} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \|\boldsymbol{\alpha}\|_1$$

has a closed form solution

$$\alpha_j = \mathbf{d}_j^T \mathbf{y}$$

But, if the dictionary is overcomplete, and consequently cannot possibly be orthonormal, we cannot guarantee that this update is optimal.

Matching pursuit – dictionary coherence

Recall that orthonomal dictionaries are those for which $\mathbf{d}_i^T \mathbf{d}_j = 0, i \neq j$.

Departure from orthonomality called coherence is defined as

$$\mathcal{M}(\mathbf{D}) = \max_{i,j} |\mathbf{d}_i^T \mathbf{d}_j|.$$

If coherence is small then update

$$\alpha_j = \mathbf{d}_j^T \mathbf{y}$$

might still be approximately correct.

Matching pursuit algorithm [2] is based on this idea

Initialize: $\alpha_i = 0, \forall i, \mathbf{r} = \mathbf{y}$
while $\|\mathbf{r}\|_2 > \text{threshold}$ **do**
  $k = \text{argmax}_j |\mathbf{d}_j^T \mathbf{r}|$
  $\alpha_k = \mathbf{d}_k^T \mathbf{r}$
  $\mathbf{r} = \mathbf{r} - \alpha_k \mathbf{d}_k$
end while

Learning dictionaries

In order to make story simpler we will assume that there is a patchset[1] $\mathrm{Data} = \{\mathbf{y}_i : i = 1..n\}$

$$\underset{\boldsymbol{\alpha}, \mathbf{D}}{\text{minimize}} \frac{1}{2} \sum_i \|\mathbf{y}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 + \lambda \|\boldsymbol{\alpha}_i\|_1$$

We want to simultaneously learn weights for each patch and dictionary.

[1] We do not need to bother with details of patch extraction, since we are not focusing on reconstruction of clean signal. We did that last time.

Learning dictionaries

A problem

$$\mathbf{D}\boldsymbol{\alpha}_i = (\frac{1}{3}\mathbf{D})(3\boldsymbol{\alpha}_i)$$

so we could learn dictionary $\mathbf{D}$ and weights $\boldsymbol{\alpha}_i$ or just scale the dictionary down and weights up.

More broadly, matrix factorization $\mathbf{A} = \mathbf{BC}$ is not unique[2]

For some $\mathbf{T}$ such that $\mathbf{T}\mathbf{T}^T = \mathbf{I}$ we can obtain a different factorization $\mathbf{A} = (\mathbf{BT})(\mathbf{T}^T\mathbf{C})$.

Q: What could we possibly do?

[2] Be on the lookout for such problems when your optimization problem has product terms.

Learning dictionaries

Just constrain the dictionary elements to have norm less or equal to 1

$$\begin{aligned} \underset{\boldsymbol{\alpha}, \mathbf{D}}{\text{minimize}} \quad & \frac{1}{2} \sum_i \|\mathbf{y}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_1 \\ \text{subject to} \quad & \|\mathbf{d}_i\|_2 \leq 1, \forall i \end{aligned}$$

Q: Can you argue that with this constraint optimal dictionary consists of elements with norm exactly 1?

Learning dictionaries

$$\begin{aligned} \underset{\boldsymbol{\alpha}, \mathbf{D}}{\text{minimize}} \quad & \frac{1}{2} \sum_i \|\mathbf{y}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_1 \\ \text{subject to} \quad & \|\mathbf{d}_j\|_2 \leq 1, \forall j \end{aligned}$$

Even though we eliminated spurious weight and dictionary combinations with unit norm constraint a more insidious problem persists.

There exist local minima, solutions that locally look optimal but are not global optima.

The problem is convex in $\boldsymbol{\alpha}$ *given* $\mathbf{D}$ and vice versa.

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{argmin}} \quad & \frac{1}{2} \sum_i \|\mathbf{y}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_1 \\ \underset{\mathbf{D}, \forall j \|\mathbf{d}_j\|_2 \leq 1}{\text{argmin}} \quad & \frac{1}{2} \sum_i \|\mathbf{y}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_1 \end{aligned}$$

Learning dictionaries – an algorithm

for $t = 1, \ldots \mathrm{MAXITERATIONS}$ **do**
  for $i = 1, \ldots T$ **do**
    $$\boldsymbol{\alpha}_i = \underset{\boldsymbol{\alpha}_i}{\operatorname{argmin}} \frac{1}{2} \left\| \mathbf{y}_i - \mathbf{D}\boldsymbol{\alpha}_i \right\|_2^2 + \lambda \left\| \boldsymbol{\alpha}_i \right\|_1$$
  end for
  $$\mathbf{D} = \underset{\mathbf{D}, \forall j \left\| \mathbf{d}_j \right\|_2 \leq 1}{\operatorname{argmin}} \frac{1}{2} \sum_i \left\| \mathbf{y}_i - \mathbf{D}\boldsymbol{\alpha}_i \right\|_2^2$$
end for

Updating dictionaries

A number of different algorithms to solve

$$\boldsymbol{\alpha}_i = \underset{\boldsymbol{\alpha}_i}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{y}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 + \lambda \|\boldsymbol{\alpha}_i\|_1$$

- Coordinate descent
- Matching Pursuit
- LARS (Least Angle Regression)[3]
- FISTA (Fast Iterative Shrinkage and Thresholding Algorithm)[1].

What about dictionary update

$$\mathbf{D} = \underset{\mathbf{D}, \forall j \left\|\mathbf{d}_j\right\|_2 \leq 1}{\operatorname{argmin}} \frac{1}{2} \sum_i \|\mathbf{y}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 ?$$

We will use *projected* gradient descent.

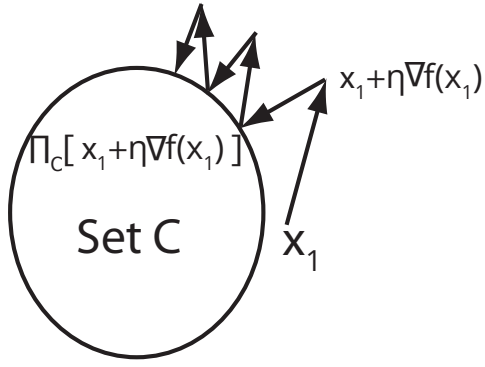
Projected gradient descent

Like gradient descent, but with projection back onto the constrained set.

$$\underset{\mathbf{x} \in C}{\text{minimize}}\, f(\mathbf{x})$$

The algorithm alternates two steps:

for $i = 1, .., \text{MAXIT}$ **do**
  $\mathbf{x}^0 = \mathbf{x} - \eta \nabla f(\mathbf{x})$
  $\mathbf{x} = \Pi_C[\mathbf{x}^0]$
end for



Projection step for unit norm ball

Projection is defined as

$$\Pi_C[\mathbf{x}] = \operatorname*{argmin}_{\mathbf{z} \in C} \|\mathbf{x} - \mathbf{z}\|_2^2$$

a point in set C closest to point $\mathbf{x}$.

In our case for a single column of dictionary $S = \{\mathbf{z} : \|\mathbf{z}\|_2 \leq 1\}$ and

$$\Pi_S[\mathbf{x}] = \frac{\mathbf{x}}{\max(1, \|\mathbf{x}\|_2)}$$

and it amounts to normalizing a vector only if its norm is greater than 1.

For full dictionary $C = \{\mathbf{A} : \forall j \, \|\mathbf{a}_j\|_2 \leq 1\}$ and

$$\Pi_C[\mathbf{D}] = \begin{bmatrix} \Pi_S[\mathbf{d}_1] & \Pi_S[\mathbf{d}_2] & \cdots & \Pi_S[\mathbf{d}_1] \end{bmatrix}$$

So, just normalize columns whose norm is larger than 1.

Dictionary learning update

Hence a single dictionary update, for given $\boldsymbol{\alpha}$ amounts to running this code

for $i = 1, .., \mathrm{MAXIT}$ **do**
$\mathbf{D}^0 = \mathbf{D} + \frac{\eta}{n} \sum_i (\mathbf{y}_i + \sum_i (\mathbf{y}_i - \mathbf{D}\alpha_i)\alpha_i^T$
$\mathbf{D} = \Pi_C[\mathbf{D}^0]$
end for

where n is the number of samples (or patches) and η is a step size.

Full learning algorithm

for $t = 1, ...\text{MAXITERATIONS}$ **do**
  for $i = 1, ...T$ **do**
    $\boldsymbol{\alpha}_i = \underset{\boldsymbol{\alpha}_i}{\text{argmin}} \frac{1}{2} \|\mathbf{y}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 + \lambda \|\boldsymbol{\alpha}_i\|_1$
  end for
  for $i = 1, .., \text{MAXIT}$ **do**
    $\mathbf{D}^0 = \mathbf{D} + \frac{\eta}{n} \sum_i (\mathbf{y}_i + \sum_i (\mathbf{y}_i - \mathbf{D}\alpha_i)\alpha_i^T$
    $\mathbf{D} = \Pi_C[\mathbf{D}^0]$
  end for
end for

Illustrative applications of dictionary learning – inpainting

Illustrative applications of dictionary learning – completion

Online dictionary learning

Batch algorithms, pass through whole dataset before updating parameters

Online algorithms work on smaller subsets of data at a time.

for $t = 1, ...\mathrm{MAXITERATIONS}$ **do**

randomly choose a sample $\mathbf{z}_t$

$$\boldsymbol{\alpha}_t = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{z}_t - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_1$$

$$\mathbf{D}^t = \underset{\mathbf{D}, \forall j \|\mathbf{d}_j\|_2}{\operatorname{argmin}} \frac{1}{t} \sum_{i=1}^{t} \frac{1}{2} \sum_i \|\mathbf{z}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2$$

end for

A dictionary learning implementation

Use SPArse Modeling Software package.

It is available from:
`http://spams-devel.gforge.inria.fr/downloads.html`

It as interfaces for R, python and Matlab.

It is fairly efficient, and you can try out your ideas in a matter of hours.

Sparse representations

We implicitly assumed that given a patch $\mathbf{y}$ and a dictionary $\mathbf{D}$ we can find a *sparse* $\boldsymbol{\alpha}$ to encode that patch so that

$$\mathbf{y} \approx \mathbf{D}\boldsymbol{\alpha}.$$

We are going to look into when we can guarantee that such sparse representation exists, and in some cases when it is guaranteed to be exact, that is

$$\mathbf{y} = \mathbf{D}\boldsymbol{\alpha}.$$

Johnson-Lindenstrauss Lemma

Lemma

For a set V points in $\mathbf{R}^n$ and some $\epsilon \in (0,1)$ and integer $k = O(1/\epsilon^2 \log |V|)$ there exists a map $f : \mathbf{R}^n \to \mathbf{R}^k$ such that for all $u, v \in V$

$$(1-\epsilon)\|u-v\|^2 \leq \|f(u)-f(v)\|^2 \leq (1+\epsilon)\|u-v\|^2$$

Johnson-Lindenstrauss Lemma

The number of dimensions needed to capture the distances between datapoints with a small distortion is logarithmic in the size of the set.

- No mention of distributions – any set of points
- No mention of the source space dimension n just the number of data points $\log |V|$

Note that the projection f need not preserve the feature identities.

A coordinate in the low dimensional space may combine apples and oranges, e.g. $1/2$*height $+$ $1/3$*weight - $1/4$*age

Practical construction of mapping[3]

We will use a simple linear mapping $f(\mathbf{u}) = \mathbf{Ax}$

First, randomly generate matrix $\mathbf{U} : k \times n$

- rows uniformly sampled basis vectors of length n
- rows sampled from a spherical Gaussian $\mathcal{N}(0, I_n)$
- each entry is randomly sampled from +1/-1

$$p(u_{i,j} = 1) = 1/2 \qquad p(u_{i,j} = -1) = 1/2$$

- each entry is randomly sampled from $\{-1, 0, +1\}$

$$p(u_{i,j} = \frac{1}{\sqrt{q}}) = \frac{1}{2q} \quad p(u_{i,j} = -\frac{1}{\sqrt{q}}) = \frac{1}{2q} \quad p(u_{i,j} = 0) = 1 - \frac{1}{q}$$

From matrix $\mathbf{U}$ construct final projection matrix

$$\mathbf{A} = \frac{1}{\sqrt{k}}\mathbf{U}$$

[3] Johnson and Lindenstrauss 1984, Indyk and Motwani 1998, Achlioptas 2003,Matoušek 2008.

Sparse mapping

The last choice ensured that there are $1 - \frac{1}{q}$ zero entries in the matrix $\mathbf{A}$.

This results in a cheap projection operation with all of the benefits of the JL lemma.

Locality sensitive hashing

In our exploration of JL lemma we came up with a scheme

$$f(\mathbf{x}) = \mathbf{A}\mathbf{x}$$

that did dimensionality reduction but preserved the distances between points.

It turns out you can go even further and construct a mapping

$$h_{\mathbf{a}}(\mathbf{x}) = \operatorname{sign}(\mathbf{a}^T\mathbf{x})$$

Each row of matrix $\mathbf{A}$ defines a single hash function.

We will denote the group of hashing functions

$$h_A : \mathbf{R}^N \to \{-1, +1\}^k$$

which produces a k long sign vector. Note that storing this vector takes k bits!

Sparse representations

The framework is very similar to the regression setup.

We are given a set of vectors $(\mathbf{x}_1, \ldots, \mathbf{x}_p)$ that form a basis of some space S or alternatively a frame.[4]

We wish to find a representation of a vector $\mathbf{y}$

$$\mathbf{y} = \sum_i w_i \mathbf{x}_i = \mathbf{X}\mathbf{w}$$

where the number of nonzero ws

$$\|\mathbf{w}\|_0 = \sum_i [w_i \neq 0]$$

is as small as possible.

[4] A frame is a set of possibly linearly dependent vectors that span a space with a constraint that there exist $0 < A \leq B < \infty$ such that $A \|\mathbf{v}\|_2 \leq \sum_i \langle \mathbf{x}_i, \mathbf{v} \rangle \leq B \|\mathbf{v}\|_2$

ℓ_0 is hard but ℓ_1 is not

Solving optimization problem (P_0)

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \|\mathbf{w}\|_0 \\ \text{subject to} \quad & \mathbf{y} = \mathbf{X}\mathbf{w} \end{aligned}$$

is NP-hard, but its convex relaxation (P_1)

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \|\mathbf{w}\|_1 \\ \text{subject to} \quad & \mathbf{y} = X\mathbf{w} \end{aligned}$$

is not. This problem is named Basis Pursuit.

Note that unlike the regression we require exact equality between $\mathbf{y}$ and its representation $\mathbf{X}\mathbf{w}$

Basis Pursuit Denoising is a lasso regression analog.

When does solving (P_1) give solution to (P_0)?

We can define coherence property for our basis/frame

$$\mathcal{M}(\mathbf{X}) = \max_{i \neq j} |\mathbf{x}_i^T \mathbf{x}_j|$$

Theorem

Suppose sparsest solution w^*

$$\|\mathbf{w}^*\|_0 < \frac{1}{2}\left(1 + \frac{1}{\mathcal{M}(\mathbf{X})}\right)$$

then solving (P_1) *gives the same solutions as the original* (P_0) *problem.*

Restricted Isometry Property

Definition

A vector $\mathbf{w} \in \mathbf{R}^n$ is k-sparse if it has at most k non-zero entries.

Definition

If for any k-sparse vector $\mathbf{w}$ it holds that

$$(1 - \delta_k) \|\mathbf{w}\|_2^2 \leq \|\mathbf{X}\mathbf{w}\|_2^2 \leq (1 + \delta_k) \|\mathbf{w}\|_2^2$$

then X has k-Restricted Isometry Property (RIP) with δ_k.

Put differently, using $\mathbf{X}$ to map $\mathbf{w}$ into a different space gives a vector of similar length as $\mathbf{w}$.

Contrast with Johnson Lindenstrauss for $f(\mathbf{u}) = \mathbf{A}\mathbf{u}$

$$(1 - \epsilon) \|\mathbf{u} - \mathbf{v}\|^2 \leq \|\mathbf{A}(\mathbf{u} - \mathbf{v})\|^2 \leq (1 + \epsilon) \|\mathbf{u} - \mathbf{v}\|^2$$

If matrix $\mathbf{A}$ satisfies JL Lemma then RIP can be shown to hold for this matrix as well.

When does solving (P_1) give solution to (P_0)?

$$(1 - \delta_k) \|\mathbf{w}\|_2^2 \leq \|\mathbf{X}\mathbf{w}\|_2^2 \leq (1 + \delta_k) \|\mathbf{w}\|_2^2$$

Theorem

If $\mathbf{X}$ *is k-RIP with* $\delta_{2k} < \sqrt{2} - 1$*, then for all k-sparse vectors* $\mathbf{w}$ *such that* $\mathbf{X}\mathbf{w} = \mathbf{y}$ *the solution for* (P_1) *is equal to solution of* (P_0)*.*

To summarize, if you construct a matrix that is k-RIP then you can represent a vector $\mathbf{y}$ in this basis with some small distortion. Further, recovery of this representation can be done solving a convex optimization problem (P1).

Sensing

If a high-dimensional signal $\mathbf{y}$ is sparsely representable in some basis $\mathbf{X}$

$$\mathbf{y} = \mathbf{X}\mathbf{w}$$

how much of $\mathbf{y}$ do we need to know to reconstruct $\mathbf{w}$ and hence the rest of $\mathbf{y}$.

We will split the issues of representation and sensing:

- representation: $\mathbf{y} = \mathbf{X}\mathbf{w}$ where w is k-sparse vector
- sensing: $\mathbf{z} = \mathbf{R}\mathbf{y} = \mathbf{R}\mathbf{X}\mathbf{w}$ is measured

Sensing

Assuming we only measured some entries of vector $\mathbf{z} = \mathbf{R}\mathbf{y} = \mathbf{R}\mathbf{X}\mathbf{w}$ indexed by set M the recovery problem is

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \|\mathbf{w}\|_1 \\ \text{subject to} \quad & z_i = \langle \mathbf{r}_i, \mathbf{X}\mathbf{w} \rangle, \forall i \in M \end{aligned}$$

How large does set M have to be?

Recovery from compressive sensing measurements

We define coherence between basis

$$\mu(X, R) = \sqrt{n} \max_{i,j} | \langle \mathbf{x}_i, \mathbf{r}_j \rangle |$$

and for any two bases $\mu(\mathbf{A}, \mathbf{B}) \in [1, \sqrt{n}]$

Theorem

Given a representation basis $\mathbf{X}$ *and sensing basis* $\mathbf{R}$ *and a vector* $\mathbf{y} = \mathbf{X}\mathbf{w}$ *with* $\mathbf{w}$ *k-sparse then if*

$$m \geq C \mu^2(\mathbf{X}, \mathbf{R}) k \log \frac{n}{\delta}$$

and we randomly select m values to sense from $\mathbf{z} = \mathbf{R}\mathbf{y}$ *we can recover* $\mathbf{w}$ *with probability* $1 - \delta$.

Practicalities

You don't need to know the k up front, you simply solve the problem

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \|\mathbf{w}\|_1 \\ \text{subject to} \quad & z_i = \langle \mathbf{r}_i, \mathbf{X}\mathbf{w} \rangle, \forall i \in M \end{aligned}$$

The choice of M is random so any subset of vectors in $\mathbf{R}$ is fine.

Note that the problem is not the same as Lasso but you can derive an ADMM algorithm for example.

Compressive sensing further reading

Compressive sensing is a very active field.

Your best bet in exploring applications is to start at `http://dsp.rice.edu/cs`

Interesting ideas for projects relating to compressive sensing would involve learning dictionaries using prior knowledge and possibly giving strong theoretical guarantees about ability to reconstruct signals.


Amir Beck and Marc Teboulle.
A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems.
SIAM J. Img. Sci., 2(1):183–202, March 2009.


S.G. Mallat and Z. Zhang.
Matching pursuits with time-frequency dictionaries.
Signal Processing, IEEE Transactions on, 41(12):3397–3415, Dec 1993.


Bradley Efron Trevor, Trevor Hastie, Lain Johnstone, and Robert Tibshirani.
Least angle regression.
Annals of Statistics, 32:407–499, 2002.