دانشگاه صنعتی شریف

دانشکده مهندسی برق

پایان‌نامه کارشناسی ارشد

گرایش مخابرات

نمایش تُنُک و کاربرد آن در نویززدائی تصاویر

نگارنده

مصطفی صادقی

استاد راهنما

دکتر مسعود بابائی‌زاده

مهرماه ۱۳۹۱


بسم الله الرحمن الرحيم

بسمه تعالی

دانشگاه صنعتی شریف
دانشکده مهندسی برق

پایان‌نامه کارشناسی ارشد

عنوان: **نمایش تُنُک و کاربرد آن در نویززدائی تصاویر**

نگارش: «**مصطفی صادقی**»

اعضاء هیأت داوران:

دکتر مسعود بابائی‌زاده امضاء:.....

دکتر حمید سلطانیان‌زاده امضاء:.....

دکتر بابک حسین‌خلج امضاء:.....

تاریخ: ۱۱ مهر ۱۳۹۱.

قدردانی

در ابتدا خداوند متعال را شاکرم که توفیق یادگیری و فهم علم و دانش را به من عنایت فرمود و این توانائی را در من بوجود آورد تا بتوانم در راه پژوهش گام بردارم ...

از استاد گرانقدرم، آقای دکتر بابائی‌زاده، که راهنمائی‌ها و نکته‌های ارزنده‌ی ایشان راه‌گشائی در جهت به ثمر رسیدن این پایان‌نامه بود، صمیمانه سپاسگزارم. هم‌چنین از اساتید گرامی، آقایان دکتر سلطانیان‌زاده و دکتر خلج، که زحمت حضور در جلسه‌ی دفاع را پذیرفتند، کمال تشکر را دارم.

تقدیم به ...

روح پاک مادرم؛

دریای بی کران فداکاری و عشق که وجودم برایش همه رنج بود و وجودش برایم همه مهر

و

پدرم؛

که عالمانه به من آموخت تا چگونه در عرصه زندگی، ایستادگی را تجربه نمایم

چکیده:

پردازش تُنُک سیگنال‌ها به عنوان ابزاری قدرتمند و جایگزینی کارآ برای تبدیلات کلاسیک کامل طی دهه‌ی اخیر به شدّت مورد توجه قرار گرفته است. در این رهیافت می‌خواهیم از بین تعَداد زیادی سیگنال پایه، که در حالت کلّی تعدادشان خیلی بیشتر از بُعدشان است، کم‌ترین تعداد را برای نمایش یک سیگنال انتخاب کنیم. هر سیگنال پایه یک «اتم» و مجموعه‌ی این اتم‌ها یک «دیکشنری» نامیده می‌شود. این عمل در حالت کلی دشوار بوده و جزء مسائل NP-hard است؛ چرا که نیازمند یک جستجوی ترکیبیاتی است. در سال‌های اخیر امّا با ارائه‌ی پُشتوانه‌های تئوریک و معرفی الگوریتم‌های عملی نشان داده شده است که تُنُک‌ترین نمایشِ یک سیگنال در یک دیکشنری فوق کامل تحت شرایطی یکتا بوده و می‌توان این جواب را در زمان محدَود بدست آورد. به این ترتیب این مبحث به سرعت در کاربردهای گوناگونِ پردازش سیگنال از جمله فشرده‌سازی داده‌ها، جداسازی کور منابع، بهبود تصاویر، تصویربرداری پزشکی، تشخیص الگو و ... مورد استفاده قرار گرفت. دو مسأله‌ی مهم در پردازش تُنُک وجود دارد. یکی از این مسائل، پیدا کردن یک دیکشنری فوق کامل مناسب برای یک کلاس مشخص از داده‌ها است؛ یعنی دیکشنری‌ای که بتواند برای همه‌ی سیگنال‌های آن کلاس، یک نمایش به اندازه‌ی کافی تُنُک ارائه دهد. این موضوع منجر به توسعه‌ی الگوریتم‌های آموزش دیکشنری شده است. مسأله‌ی دوم داشتن یک الگوریتم کارآ برای بدست آوردن تُنُک‌ترین نمایش سیگنال (یا کُدینگ تُنُک سیگنال) است. این مسأله نیز منجر به معرفی الگوریتم‌های زیادی برای این منظور شده است. در این پایان‌نامه، ابتدا مروری مختصر داریم بر تعدادی از الگوریتم‌های موجود برای کُدینگ تُنُک و آموزش دیکشنری و هم‌چنین مسأله‌ی نویززدائی از تصاویر با استفاده از نمایش تُنُک. در ادامه، ابتدا یک الگوریتم جدید برای کُدینگ تُنُک سیگنال‌ها معرفی می‌کنیم. این الگوریتم تعمیمی از الگوریتم‌های IRLS است. شبیه‌سازی‌ها حکایت از عملکرد بهتر الگوریتم معرفی شده نسبت به الگوریتم‌های IRLS معمولی دارد. در ادامه، چند الگوریتم جدید برای آموزش دیکشنری معرفی می‌کنیم. دو مورد از این الگوریتم‌ها در واقع تلفیقی از دو الگوریتم معروف MOD و K-SVD است. برای غلبه بر حجم محاسبات سنگین الگوریتم K-SVD، یک الگوریتم کارآ معرفی می‌کنیم. شبیه‌سازی‌ها روی داده‌های مختلف بیانگر برتری الگوریتم پیشنهادی نسبت به K-SVD، هم از نظر زمان اجرا و هم از نظر کیفیت جواب‌ها است. سپس سه الگوریتم جدید دیگر معرفی می‌کنیم که از نظر ساختار با الگوریتم‌های موجود متفاوت بوده و البته با توجه به نتایج شبیه‌سازی‌های مختلف، عملکرد متوسط بهتری دارند. در ادامه، برای بهبود عملکرد نویززدائی با استفاده از نمایش تُنُک، روشی معرفی می‌کنیم که براساس متوسط‌گیری از تخمین‌های مختلف بلوک‌های تصویر طی فرآیند آموزش دیکشنری است. شبیه‌سازی انجام شده نشان‌دهنده‌ی عملکرد رضایت‌بخش این روش است.

کلمات کلیدی:

۱- نمایش تُنُک Sparse Representation.
۲- حسگری فشرده Compressed Sensing.
۳- آموزش دیکشنری Dictionary Learning.
۴- نویززدائی تصویر Image Denoising.

فهرست مطالب

فهرست جداول

فهرست اشکال

فهرست کلمات اختصاری

BP	Basis Pursuit
CoSaMP	Compressive Sampling Matching Pursuit
CS	Compressed Sensing
DCT	Discrete Cosine Transform
IHT	Iterative Hard Thresholding
IRLS	Iteratively Re-weighted Least Squares
IST	Iterative Shrinkage-Thresholding
KLT	Karhunen-Loève Transform
MAP	Maximum A Posteriori Probability
ML	Maximum Likelihood
MM	Majorization Minimization
MOD	Method of Optimal Directions
MP	Matching Pursuit
NP	Non Polynomial
OMP	Orthogonal Matching Pursuit
PCA	Principal Component Analysis
RLS	Recursive Least Squares
SL0	Smoothed $\ell_\circ$ (norm)
SNR	Signal to Noise Ratio
StOMP	Stagewise Orthogonal Matching Pursuit
SVD	Singular Value Decomposition
USLE	Underdetermined System of Linear Equations
UoS	Union of Subspaces
VQ	Vector Quantization

فصل ۱

مقدمه

نمایش سیگنال‌ها در یک پایه‌ی[۱] مناسب همواره به عنوان یک گام اساسی در شناخت ویژگی‌ها و نیز استخراج و تفسیر اطّلاعات سیگنال‌ها مورد توجه بوده است [۵۷]. تبدیل‌های کلاسیکِ متعامد یکّه[۲] به دلیل سادگی محاسبات و بعلاوه، داشتن جوابی یکتا برای نمایش سیگنال‌ها، در جامعه پردازش سیگنال مورد توجه قرار داشته و دارد. از این دست تبدیل‌ها می‌توان به تبدیل فوریه، تبدیل کسینوسی گسسته و تبدیل ویولت اشاره کرد. این دسته از تبدیل‌ها اصطلاحاً «کامل» هستند؛ یعنی تعداد سیگنال‌های پایه با بُعد سیگنال برابر است. عیبی که این تبدیل‌ها دارند این است که فقط برای نمایش دسته‌ی محدودی از سیگنال‌ها مناسب هستند. یک تبدیل و یا نمایشِ «مناسب» این ویژگی را دارد که برای ساختن یا بسط سیگنال با یک خطای مشخص، تنها از تعداد کمی سیگنال پایه استفاده می‌شود. این همان مفهوم «سادگی نمایشِ» سیگنال‌ها است؛ چرا که ما همواره به دنبال ساده‌ترین نمایش برای سیگنال تحت بررسی خود هستیم که با کم‌ترین تعداد سیگنال پایه توصیف شود.

بُردار $\mathbf{y} \in \mathbb{R}^n$ را در نظر بگیرید. می‌خواهیم نمایش این بُردار را بر حسب سیگنال‌های پایه‌ی $\{\mathbf{d}_i\}_{i=1}^m$ بدست آوریم. به بیان ریاضی، می‌خواهیم بُردار $\mathbf{x} \in \mathbb{R}^m$ را از دستگاه معادلات خطی زیر بدست آوریم:

$$\mathbf{y} = \sum_{i=1}^{m} x_i \mathbf{d}_i = \mathbf{D}\mathbf{x},$$

که در آن، $\mathbf{D} \in \mathbb{R}^{n \times m}$ ماتریس تبدیل بوده که ستون‌های آن همان سیگنال‌های پایه هستند. زمانیکه $m = n$ و

[۱] Basis

[۲] Orthonormal

خصوصاً ماتریس $\mathbf{D}$ متعامد یکّه باشد، جواب دستگاه فوق یکتا بوده و به راحتی از رابطه‌ی زیر بدست می‌آید:

$$\mathbf{x} = \mathbf{D}^{-1}\mathbf{y} = \mathbf{D}^T\mathbf{y}.$$

همانطور که قبلاً ذکر شد، در این وضعیت اگرچه جوابی یکتا برای نمایش سیگنال بدست می‌آید، امّا در بسیاری از موارد این جواب تُنُک نیست. به عنوان مثال، پایه‌ی فوریه فقط برای سیگنال‌های سینوسی یا سیگنال‌های هموارِ تناوبی مناسب است (یعنی جواب تُنُکی دارد)، در حالی که برای سیگنال‌هائی که گذرا بوده یا شامل ضربه هستند جواب تُنُکی ندارد. به عبارت بهتر، هر پایه برای یک کلاس محدود از داده‌ها مناسب است.

ایده‌ی پردازش تُنُک این است که تعداد سیگنال‌های پایه را (خیلی) بیشتر از بُعد سیگنال بگیریم؛ یعنی طبق نمادگذاری قبل، $m > n$. در این حالت، دستگاه معادلات خطی بدست آمده فرومعین[۱] است، چرا که تعداد معادلات از تعداد مجهولات کمتر است. در این وضعیت، دستگاه جواب ندارد اگر $\mathbf{y}$ در گستره‌ی[۲] ستون‌های ماتریس $\mathbf{D}$ نباشد و در غیر اینصورت بی‌شمار جواب دارد. البته فرض ما بر این است که ماتریس $\mathbf{D}$ رُتبه‌ی کامل است، در نتیجه حداقل یک پایه در میان ستون‌های آن وجود دارد. لازم به ذکر است که در بحث پردازش تُنُک، ماتریس $\mathbf{D}$ یک «دیکشنری» و هر ستون از آن یک «اتم» نامیده می‌شود. این نامگذاری پس از معرفی دیدگاه تجزیه‌ی اتمی[۳] اتخاذ شد [۴۵].

در قلب بحث کُدینگ تُنُک سیگنال‌ها، یک دستگاه معادلات خطی فرومعین یا به اختصار USLE[۴]، به ترتیبی که دیدیم، قرار دارد. برای بیان دقیق‌تر کُدینگ تُنُک، به یاد آورید که ما از ابتدا به دنبال ساده‌ترین نمایش برای یک سیگنال بودیم؛ نمایشی که در آن کم‌ترین تعداد اتم، سیگنال مورد بررسی ما را می‌سازند. بنابراین طبیعی به نظر می‌رسد که ما از بین بی‌شمار جواب USLE، تُنُک‌ترین جواب ممکن را انتخاب کنیم. امّا این هنوز شروع کار است. سؤالات زیادی وجود دارد که باید پاسخ داده شود. اول اینکه آیا اصولاً تُنُک‌ترین جواب این دستگاه معادلات یکتا است؟ چرا که اگر چندین جواب متمایز به عنوان تُنُک‌ترین نمایش یا ساده‌ترین توصیف برای سیگنال مورد بررسی ما وجود داشته باشد، برای ما سودی ندارد. دوم اینکه، اگر تُنُک‌ترین نمایش سیگنال یکتا باشد، با چه الگوریتمی این جواب را به دست آوریم؟ یا آیا اصلاً الگوریتمی وجود دارد که در زمان محدود و با حجم محاسبات نه چندان

[۱] Underdetermined

[۲] Span

[۳] Atomic Decomposition

[۴] Underdetermined System of Linear Equations

بالا این جواب را پیدا کند؟ در پاسخ باید گفت که انبوه مقالاتی که در زمینه پردازش تُنُک سیگنال‌ها طی دهه‌ی اخیر منتشر شده است نشان می‌دهد که اولاً تُنُک ترین جواب USLE تحت شرایطی یکتا بوده و ثانیاً الگوریتم‌های بسیار کارآئی برای پیدا کردن این جواب وجود دارد؛ بر خلاف نگاه اول که حل این دستگاه معادلات را مستلزم جستجوی ترکیبیاتی می‌بیند (به عنوان یک مرجع خوب در این زمینه [۲۴] را ببینید).

امّا اهمیت نمایش تُنُک چیست؟ برای پاسخ به این سوال، ابتدا اهمیت داشتن یک «مدل» برای سیگنال‌ها را بیان می‌کنیم. تقریباً تمامی کاربردهای پردازش سیگنال مبتنی بر اتخاذ یک مدل مناسب برای سیگنال‌ها است [۱۱، ۲۶]. مدل، در حقیقت مشخصات سیگنال را توضیح می‌دهد و به نوعی هویت یک سیگنال است. در مسائل خطی معکوس[۱]، داشتن یک مدل برای سیگنال‌ها از اهمیت زیادی برخوردار است. به عنوان یک مثال نسبتاً ساده (که البته از مطالب مورد بررسی این پایان‌نامه نیز هست)، مسأله‌ی نویززدائی از سیگنال‌ها (در اینجا تصاویر) را در نظر بگیرید. مسأله به این قرار است که ما یک نسخه‌ی آغشته به نویز (جمع شونده) از یک سیگنال را در اختیار داریم و می‌خواهیم سیگنال تمیز مطلوب را بازیابی کنیم. این کار مستلزم این است که یک ساختار (یا مدل) برای سیگنالِ مطلوب فرض کنیم؛ این مدل می‌تواند به اینصورت باشد که سیگنالِ ما «هموار» است، به این ترتیب سیگنال تمیز از نویز که ناهموار است متمایز می‌شود.

در بحث پردازش تُنُک، مدلی که برای سیگنال در نظر می‌گیریم، اجتماعی از زیرفضاها[۲] یا به اختصار UoS است [۵، ۶، ۲۸]. برای روشن‌تر شدن موضوع، فرض کنید که سیگنال مورد بررسی، ما یعنی $\mathbf{y}$، تنها از s اتم استفاده می‌کند که $s \ll m$. بنابراین برای ساختن این سیگنال به تعداد $N = \binom{m}{s}$ دسته‌ی s تائی از اتم‌ها حق انتخاب داریم. هر انتخاب در واقع نشان‌دهنده‌ی یک «زیرفضا» است و با در نظر گرفتن تمام این زیر فضاها، مدل سیگنال ما می‌شود اجتماعی از زیر فضاها. به بیان ریاضی داریم:

$$\mathbf{y} \in \mathcal{S} = \bigcup_{i=1}^{N} \mathcal{S}_i,$$

که $\mathcal{S}_i$ ها زیرفضاهائی s بُعدی از $\mathbb{R}^n$، ساخته شده با استفاده از اتم‌های دیکشنری هستند. توجه کنید که UoS یک مدلِ «غیرخطی» است؛ چرا که اگر دو سیگنال در این مدل صدق کنند لزوماً هر ترکیبِ خطی از آن دو در این مدل صدق نخواهد کرد. برگردیم به مثال نویززدائی. با فرض مدل UoS برای سیگنالِ تمیز، در حقیقت فرض کرده‌ایم

[۱] Linear Inverse Problem

[۲] Union of Subspaces

که سیگنالِ مطلوب ما در یک دیکشنریِ مشخص نمایشی تُنُک دارد. همین نکته وجه تمایز سیگنال تمیز و نویز است؛ چرا که دیکشنری مخصوصِ کلاسِ سیگنال‌های (طبیعی) مطلوب ما است و یا به بیان دیگر، نویز در این دیکشنری نمایش تُنُکی ندارد.

در انتهای پاراگراف قبل نکته‌ای را به طور ضمنی بیان کردیم و آن «دیکشنریِ مناسب» برای یک کلاس مشخص از سیگنال‌ها است. منظور از یک «دیکشنریِ مناسب» آنی است که برای تقریباً همه‌ی سیگنال‌های یک کلاس، نمایشی به اندازه‌ی کافی تُنُک ارائه دهد. امّا سوال اساسی اینجاست که برای یک کلاس مشخص از سیگنال‌ها چنین دیکشنری‌ای را چطور پیدا کنیم؟ در پاسخ باید گفت که به طور کلی دو راه برای این کار وجود دارد. یکی استفاده از دیکشنری‌های از پیش ساخته شده است. مثلاً دیکشنری DCT فوقِ کامل[۱] برای کلاسِ تصاویر طبیعی، یا دیکشنری Gabor برای سیگنال‌های صحبت. این دسته به «دیکشنری‌های غیر وفقی»[۲] معروف هستند. دسته‌ی دوم، که البته از دسته‌ی اول کارآتراند، موسوم به «دیکشنری‌های وفقی» بوده که با استفاده از تعدادی داده‌ی آموزشی[۳] از کلاسِ مورد نظر «آموزش» می‌بینند [۴۲، ۴۰، ۵۴، ۲۹].

یکی از کاربردهای مهم پردازش تُنُک سیگنال‌ها که مورد توجه زیادی قرار گرفته است، حسگری فشرده[۴] یا به اختصار CS است [۱۰، ۱۸]. اساس حسگری فشرده بر این اصل استوار است که اگر یک سیگنال در پایه‌ای نمایشی تُنُک داشته باشد، آنگاه می‌توان با تعدادی اندازه‌گیری (تصادفی) از آن (که در حالت کلی خیلی کمتر از طول سیگنال است) این سیگنال را بازیابی کرد. این تکنیک به نوعی در تقابل با نظریه‌ی کلاسیک نمونه‌برداری شانون–نایکوئیست[۵] قرار دارد؛ چرا که نظریه‌ی شانون–نایکوئست بیان می‌کند که برای بازسازی کامل یک سیگنال، نرخ نمونه‌برداری باید حداقل به اندازه‌ی دو برابرِ حداکثر فرکانس موجود در آن باشد؛ حال آنکه با استفاده از حسگری فشرده، به تعداد نمونه (در اینجا، اندازه‌گیری)های خیلی کمتری نیاز است.

الگوریتم‌های زیادی برای کُدینگ تُنُک سیگنال‌ها و نیز آموزش دیکشنری طی چند سال اخیر ارائه شده است و البته هنوز هم الگوریتم‌های جدیدی معرفی می‌شود. با توجه به گستردگی زیاد این دو زمینه (یعنی کُدینگ

[۱] درایه‌های یک دیکشنری DCT دو بُعدی عبارتند از $d_{ij} = \cos((i-1)(j-1)\frac{\pi}{m})$, $1 \le i \le n$, $1 \le j \le m$. برای یک دیکشنری DCT فوق کامل داریم $m > n$.

[۲] Non-adaptive

[۳] Training Data

[۴] Compressed Sensing

[۵] Shannon-Nyquist

تُنُک و آموزش دیکشنری)، هر کدام به طور جداگانه می‌تواند موضوع یک پایان‌نامه باشد. ما در این پایان‌نامه به طور خیلی مختصر چند الگوریتم معروف در این دو زمینه را مرور کرده، کاربرد پردازش تُنُک سیگنال‌ها را در نویززدائی تصاویر بررسی می‌کنیم. سپس چند الگوریتم جدید، هم برای کُدینگ تُنُک و هم برای آموزش دیکشنری ارائه می‌دهیم و با انجام شبیه‌سازی‌های متعدّد، کارآئی این الگوریتم‌ها را نسبت به الگوریتم‌های دیگر نشان می‌دهیم.

شیوه‌ی ارائه‌ی مطالب به این ترتیب است. در فصل دوم، مباحث پایه‌ای بحث نمایش تُنُک را به همراه تعدادی از الگوریتم‌هائی که تاکنون برای بازیابی نمایش تُنُک سیگنال‌ها ارائه شده است، به اختصار مرور می‌کنیم. این مباحث شامل قضایای یکتائی و پایداری تُنُک‌ترین نمایش یک سیگنال است. در انتهای این فصل، به عنوان کاربردی از پردازش تُنُک، مروری مختصر خواهیم داشت بر بحث حسگری فشرده. در فصل سوم، بحث آموزش دیکشنری را مطرح خواهیم کرد. در این فصل ابتدا فُرم کلّی مسأله‌ی آموزش دیکشنری را معرفی کرده، سپس قضیه‌ی یکتائی دیکشنری را بیان می‌کنیم. در ادامه، تعدادی از الگوریتم‌های آموزش دیکشنری را که طی چند سال گذشته معرفی شده است، به اختصار بررسی می‌کنیم. در فصل چهارم، نویززدائی از تصاویر را با استفاده از نمایش تُنُک بررسی می‌کنیم. دو فصل ۵ و ۶ اختصاص به الگوریتم‌های پیشنهادی این پایان‌نامه دارد. در فصل پنجم، یک الگوریتم جدید برای بازیابی نمایش تُنُک سیگنال‌ها پیشنهاد می‌کنیم. در فصل ششم نیز، تعدادی الگوریتم جدید برای آموزش دیکشنری پیشنهاد خواهیم کرد. نهایتاً، در فصل هفتم به جمع‌بندی و نتیجه‌گیری مطالب ارائه شده و ارائه‌ی پیشنهاداتی برای کارهای آینده خواهیم پرداخت.

فصل ۲

نمایش تُنُک سیگنال‌ها، مقدمه‌ای بر تئوری و الگوریتم‌ها

۲-۱ مقدمه

در این فصل مسائل تئوری نمایش تُنُک را به همراه تعدادی از الگوریتم‌های موجود مرور می‌کنیم. ابتدا دستگاه معادلات خطی فرومعین را به عنوان اساس بحث نمایش تُنُک معرفی می‌کنیم. سپس قضایای یکتائی تُنُک‌ترین نمایش یک سیگنال را بیان کرده و در ادامه پایداری تقریب تُنُک سیگنال‌ها را بررسی می‌کنیم. تعدادی از الگوریتم‌های بدست‌آوردن نمایش تُنُک سیگنال‌ها را که تاکنون معرفی شده‌اند به اختصار مرور می‌کنیم. در نهایت، حسگری فشرده را به عنوان کاربردی از نمایش تُنُک و تحوّلی در نمونه‌برداری و فشرده‌سازی سیگنال‌ها به طور مختصر معرفی می‌کنیم. مرجع اصلی این فصل، کتاب [۲۴] است که به شیوه‌ای مناسب مباحث تئوری، الگوریتم‌ها و کاربردهای نمایش تُنُک در پردازش تصویر را پوشش داده است.

۲-۲ دستگاه معادلات خطی فرومعین

همانطور که در فصل ۱ دیدیم، در قلب پردازش تُنُک سیگنال‌ها یک دستگاه معادلات خطی فرومعین قرار دارد. ماتریس $\mathbf{D} \in \mathbb{R}^{n \times m}$ را که $m > n$ در نظر بگیرید. در این صورت دستگاه معادلات خطی $\mathbf{y} = \mathbf{D}\mathbf{x}$ فرومعین بوده و جواب ندارد اگر $\mathbf{y}$ در گستره ستون‌های ماتریس $\mathbf{D}$ نباشد و در غیر اینصورت بی‌شمار جواب دارد. برای

جلوگیری از بروز حالت نامطلوب عدم وجود جواب برای $\mathbf{x}$ ، فرض ما بر این است که ماتریس $\mathbf{D}$ رُتبه کامل است. در مسائل مختلف پردازش سیگنال به چنین دستگاه معادلات خطی فرومعینی برمی‌خوریم؛ به عنوان مثال، بسیاری از مسائل معکوس در این دسته قرار دارند. در این گونه مسائل، ما به دنبال یافتن سیگنال مطلوبِ $\mathbf{x}$ هستیم که تحت تبدیل خطی $\mathbf{D}$ تغییر شکل یافته و ما تنها نسخه‌ی با رزولوشن پائین و احتمالاً تخریب شده‌ی آن، یعنی $\mathbf{y}$، را در اختیار داریم.

گفتیم که معادله‌ی خطی فرومعین ذکر شده بی‌شمار جواب دارد. امّا آنچه برای ما مهم است تنها یکی از این جواب‌ها است. بسته به کاربرد می‌توان جوابی را انتخاب کرد که یک ویژگی منحصر به فرد داشته باشد. به عنوان مثال، جواب با حداقلِ انرژی. انتخاب یک جواب با ویژگی خاص همان رگولاریزه کردن این دستگاه معادلات است. به عبارت دقیق‌تر، اگر تابع $J : \mathbb{R}^m \longmapsto \mathbb{R}$ تشویق کننده‌ی ویژگی مورد نظر ما باشد، برای بدست آوردن این جواب باید مسأله‌ی زیر را حل کنیم:

$$P_J : \quad \min_{\mathbf{x}} J(\mathbf{x}) \quad \text{subject to} \quad \mathbf{y} = \mathbf{D}\mathbf{x}. \tag{۱-۲}$$

در مثال جوابِ با حداقل انرژی داریم $J(\mathbf{x}) = \|\mathbf{x}\|_۲^۲$. این همان رگولاریزیشن کلاسیک معروف تیخونوف[۱] است. توجه کنید که در این حالت مسئله محدّب بوده و در نتیجه این جواب یکتا است. در بسیاری از کاربردها امّا مطلوب ما جوابی است که تا حد ممکن درایه‌ی صفر (یا خیلی نزدیک صفر) داشته باشد. از جمله‌ی این کاربردها می‌توان به فشرده‌سازی داده‌ها اشاره کرد. به علاوه، بسیاری از سیگنال‌های طبیعی از جمله تصویر و سیگنال صحبت در یک دیکشنری مناسب نمایشی تُنُک دارند؛ به عبارت دیگر بُعدِ واقعی آن‌ها خیلی کمتر از بُعدِ ظاهری (تعداد مؤلفه‌ها در نمایش بُرداری) آن‌ها است. این خود می‌تواند نشانه‌ای مناسب برای بازیابی این سیگنال‌ها در مسائل معکوس از جمله نویززدائی باشد.

برای بدست آوردن تُنُک‌ترین جواب، باید مسأله‌ی زیر را حل کنیم:

$$P_۰ : \quad \min_{\mathbf{x}} \|\mathbf{x}\|_۰ \quad \text{subject to} \quad \mathbf{y} = \mathbf{D}\mathbf{x}, \tag{۲-۲}$$

که در آن $\|\mathbf{x}\|_۰ \triangleq |\text{supp}(\mathbf{x}) = \{i : \ x_i \neq ۰\}|$ نشان‌دهنده‌ی تعداد درایه‌های غیر صفر $\mathbf{x}$ است. مسأله‌ی (۲-۲) غیر محدّب است و از طرفی تابع $\|\mathbf{x}\|_۰$ ناهموار و مشتق‌ناپذیر بوده و علاوه بر این، حل این مسأله مستلزم یک جستجوی ترکیبیاتی است که در ابعاد بالا امکان‌پذیر نیست. به عبارت دیگر، برای پیدا کردن جوابی با s درایه‌ی

[۱] Tykhonov

غیر صفر، لازم است در بدترین حالت تعداد $N = \binom{m}{s}$ زیر ماتریس از $\mathbf{D}$ را، که هر کدام نماینده‌ی یک زیرفضا است، بررسی کنیم. یکی از جایگزین‌های جالب برای این تابع، نُرمِ یک است که محدّب بوده و مسأله‌ی حاصل را می‌توان با الگوریتم‌های گوناگونی و در زمان محدود حل نمود [۱۴]. تابع نُرم یک در حقیقت نزدیک‌ترین تابع محدّب به تابع شبه نُرم صفر است[۱]. در نهایت مسأله‌ی زیر را داریم:

$$P_1: \quad \min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad \text{subject to} \quad \mathbf{y} = \mathbf{D}\mathbf{x}. \tag{۲-۳}$$

به طور کلّی، یک دسته از توابع تشویق کننده‌ی تُنُک بودن بصورت $J(\mathbf{x}) = \|\mathbf{x}\|_p^p = \sum_i |x_i|^p$ با $0 \leq p \leq 1$ است. به استثنای $p = 1$ که متناظر با نُرم یک است، بقیه‌ی این توابع همه‌ی خواص یک تابع نُرم را نداشته و عموماً «شبهِ نُرم»[۲] خوانده می‌شوند. با این انتخاب، مسأله‌ی متناظر به فرم زیر است:

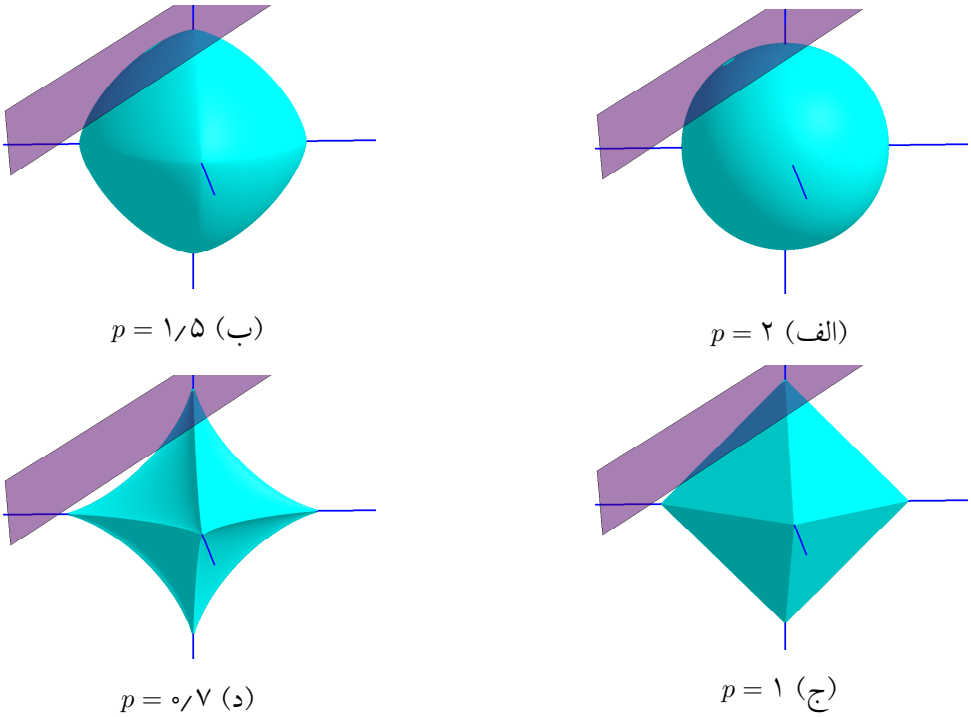
$$P_p: \quad \min_{\mathbf{x}} \|\mathbf{x}\|_p^p \quad \text{subject to} \quad \mathbf{y} = \mathbf{D}\mathbf{x}. \tag{۲-۴}$$

در ادامه یک تعبیر هندسی از تمایل این دسته از توابع به تولید جواب‌های تُنُک به نقل از [۲۴] بیان می‌کنیم. ابتدا دقّت کنید که مجموعه‌ی مجاز مسأله، یعنی دسته معادلات $\mathbf{y} = \mathbf{D}\mathbf{x}$، نشان‌دهنده‌ی یک زیرفضای آفین[۳] است. توضیح بیشتر اینکه، یک زیرفضای آفین از انتقال یک زیرفضای اقلیدسی با یک بُردار ثابت حاصل می‌شود و بنابراین لزوماً شامل مبدأ مختصات نیست. حال برای بدست آوردن جواب معادله‌ی (۲-۴) از دید هندسی، باید اصطلاحاً توپ‌های ℓ_p را، که به صورت سطوح $\|\mathbf{x}\|_p = cte.$ تعریف می‌شوند، به تدریج بزرگ کرد (به تعبیری در آن‌ها دَمید)، تا جائیکه اولین برخورد با زیرفضای آفین ذکر شده حاصل شود. برای فهم بیشتر، شکل ۲-۱ را ببینید. در این شکل، فصل مشترک زیرفضای آفین متناظر با $\mathbf{y} = \mathbf{D}\mathbf{x}$ با توپ‌های ℓ_p برای چند مقدار مختلف p نشان داده شده است. با دقّت در این شکل دو نکته را می‌توان نتیجه گرفت. اولاً، هر چه p به سمت صفر میل می‌کند، به جواب‌های تُنُک‌تری (متناظر با نقاط روی محورهای مختصات) می‌رسیم و ثانیاً، نُرم یک نیز به عنوان بهترین تقریب محدّب شبه نُرم صفر، به جواب‌های تُنُکی می‌انجامد.

[۱] ذکر این نکته ضروری است که تابع $\|.\|_1$ اگرچه محدّب است ولی اکیداً محدّب نیست؛ به عبارت دیگر، مسأله‌ی متناظر ممکن است چندین جواب مختلف امّا با نُرمِ یکِ یکسان داشته باشد. برای بحث بیش‌تر در این‌باره، به فصل اولِ [۲۴] مراجعه کنید.

[۲] Pseudo norm

[۳] Affine Subspace



شکل ۲–۱: فصل مشترک توپ‌های ℓ_p با مجموعه‌ی $\mathbf{y} = \mathbf{Dx}$ بازای چند مقدار مختلف p [۲۴].

۲–۳ یکتائی تُنُک‌ترین نمایش یک سیگنال

در این قسمت، قضایای مربوط به یکتائی تُنُک‌ترین نمایش یک سیگنال را به نقل از [۲۴] بیان می‌کنیم. ابتدا قضیه‌ی یکتائی را با استفاده از مفهوم اسپارک[۱] دیکشنری بیان می‌کنیم. اسپارک یک ماتریس روشی برای توصیف فضای پوچ[۲] آن ماتریس است. طبق تعریف، اسپارک یک ماتریس عبارت است از کمترین تعداد ستون‌هائی از آن که وابسته‌ی خطی‌اند. با این تعریف، قضیه‌ی یکتائی به صورت زیر بیان می‌شود:

قضیه ۲–۱ اگر جوابی از دستگاه فرومعینِ $\mathbf{y} = \mathbf{Dx}$ در شرط $\|\mathbf{x}\|_0 < spark(\mathbf{D})/2$ صدق کند، آنگاه لزوماً تُنُک‌ترین جواب این دستگاه معادلات خواهد بود.

همانطور که از تعریف فوق برمی‌آید، بدست آوردن اسپارک یک ماتریس خود نیازمند یک جستجوی ترکیبیاتی است. بنابراین به روش‌های ساده‌تری برای تضمین یکتائی تُنُک‌ترین جواب نیاز داریم. یک راه بسیار ساده استفاده

[۱] Spark
[۲] Null-space

از همبستگی متقابل[۱] بین اتم‌های دیکشنری است [۲۱]. طبق تعریف، همبستگی متقابل یک ماتریس عبارت است از ماکزیمم قدر مطلق ضریب همبستگی بین ستون‌های متمایز آن. به بیان ریاضی، همبستگی متقابل برای ماتریس $\mathbf{A}$ به صورت زیر تعریف می‌شود:

$$\mu(\mathbf{A}) = \max_{i \neq j} \frac{|\mathbf{a}_i^T \mathbf{a}_j|}{\|\mathbf{a}_i\|_2 \|\mathbf{a}_j\|_2} \qquad (۵-۲)$$

به این ترتیب، همبستگی متقابل، کمیّتی برای ارزیابی وابستگی خطّی بین ستون‌های یک ماتریس است. می‌توان نشان داد که رابطه‌ی بین اسپارک و همبستگی متقابل یک ماتریس بصورت $\text{spark}(\mathbf{A}) \geqslant 1 + 1/\mu(\mathbf{A})$ است [۲۴]. قضیه‌ی یکتائی با استفاده از همبستگی متقابل دیکشنری بصورت زیر بیان می‌شود:

قضیه ۲-۲ اگر جوابی از دستگاه فرومعینِ $\mathbf{y} = \mathbf{D}\mathbf{x}$ در شرط $\|\mathbf{x}\|_0 < \frac{1}{2}(1 + 1/\mu(\mathbf{D}))$ صدق کند، آنگاه لزوماً تُنُک‌ترین جواب این دستگاه معادلات خواهد بود.

دیدیم که مسأله‌ی P_0 ماهیتی ترکیباتی داشته، یک مسأله‌ی NP-hard است. به همین دلیل، مسأله‌ی P_1 که محدّب بوده و قابل حل است معرفی شد [۱۴]. نشان داده شده است که اگر جوابی از $\mathbf{y} = \mathbf{D}\mathbf{x}$ در شرط قضیه‌ی ۲-۲ صدق کند، آنگاه هم جواب مسأله‌ی P_0 است و هم جواب مسأله‌ی P_1 ؛ به عبارت دیگر، در این وضعیت این دو مسأله معادل‌اند [۱۹]. لازم به ذکر است که معمولاً $\text{spark}(\mathbf{A}) \gg 1 + 1/\mu(\mathbf{A})$ ؛ در نتیجه، شرطی که قضیه‌ی ۲-۲ روی تُنُک بودن نمایش یکتای سیگنال می‌گذارد بسیار محدودکننده‌تر از شرط قضیه‌ی ۲-۱ است. برای توضیحات بیش‌تر و نیز یک مثال عددی، [۳] را ببینید.

۲-۴ پایداری تقریب تُنُک سیگنال

تساوی $\mathbf{y} = \mathbf{D}\mathbf{x}$ عموماً در عمل برقرار نیست؛ یا لااقل اگر هم این تساوی دقیقاً بخواهد برقرار باشد لزوماً جواب‌های تُنُکی برای $\mathbf{x}$ نخواهیم داشت. در بسیاری از کاربردها، ما با تساوی $\mathbf{y} = \mathbf{D}\mathbf{x} + \mathbf{n}$ مواجه هستیم که در آن بُردار $\mathbf{n}$ با انرژی محدود، بسته به کاربرد تعابیر مختلفی دارد. به عنوان مثال در فشرده‌سازی، این بُردار نشان‌دهنده‌ی میزان اعوجاجی است که برای کُد کردن $\mathbf{y}$ می‌پذیریم. امّا در بسیاری از کاربردها، این بُردار نشان‌دهنده‌ی نویزی است که سیگنال تمیزِ $\mathbf{y}_0 = \mathbf{D}\mathbf{x}_0$ را آلوده کرده است[۲]. در این وضعیت، ما در واقع برای

[۱] Mutual-coherence

[۲] منظور از نویز در این‌جا و در ادامه‌ی این پایان‌نامه، نویز سفید گوسی جمع‌شونده است؛ مگر آنکه خلاف آن ذکر شود.

پایداری جواب، باید در اَبَرکره‌ی $\|\mathbf{y}-\mathbf{D}\mathbf{x}\|_۲ \leqslant \epsilon$ به دنبال تُنُک‌ترین جواب باشیم. برای این منظور، به جای دو مسأله‌ی $P_۰$ و $P_۱$ باید نسخه‌های پایدار نسبت به نویز آن‌ها را به ترتیب زیر حل نمائیم [۲۰]:

$$P_۰^\epsilon: \quad \min_{\mathbf{x}} \|\mathbf{x}\|_۰ \quad \text{subject to} \quad \|\mathbf{y}-\mathbf{D}\mathbf{x}\|_۲ \leqslant \epsilon, \tag{۶–۲}$$

و

$$P_۱^\epsilon: \quad \min_{\mathbf{x}} \|\mathbf{x}\|_۱ \quad \text{subject to} \quad \|\mathbf{y}-\mathbf{D}\mathbf{x}\|_۲ \leqslant \epsilon. \tag{۷–۲}$$

به این ترتیب با حلّ هر یک از مسائل فوق، می‌توانیم یک تخمین از سیگنال بدون نویزِ $\mathbf{y}_۰$ بصورت $\hat{\mathbf{y}}_۰ = \mathbf{D}\hat{\mathbf{x}}$ که $\hat{\mathbf{x}}$ جواب یکی از دو مسأله‌ی فوق است، بدست آوریم. سوال اساسی که در اینجا مطرح می‌شود این است که برای یک سطح نویز کوچک، آیا جواب تُنُکی که از حل هر یک از مسائل فوق بدست می‌آید، یعنی $\mathbf{x}_۰^\epsilon$ و $\mathbf{x}_۱^\epsilon$ ، در فاصله‌ی کمی از $\mathbf{x}_۰$ قرار دارند یا خیر؟ پاسخ مثبت به این سوال به مفهوم پایداری دو مسأله‌ی مذکور است. به بیان دیگر، اغتشاش $\boldsymbol{\Delta}\mathbf{y}$ در سیگنال و اغتشاش $\boldsymbol{\Delta}\mathbf{x}$ در نمایشِ تُنُک آن را، که رابطه‌ی بین آن‌ها بصورت $\boldsymbol{\Delta}\mathbf{y} = \mathbf{D}\boldsymbol{\Delta}\mathbf{x}$ است، در نظر بگیرید. گوئیم دو مسأله‌ی فوق (نسبت به یک اغتشاش یا نویز کوچک) پایدارند، هر گاه $\|\boldsymbol{\Delta}\mathbf{y}\|_۲$ و $\|\boldsymbol{\Delta}\mathbf{x}\|_۲$ قابل مقایسه باشند. در [۲۰] نشان داده شده است که هر دو مسأله‌ی فوق نسبت به نویز پایدارند. دو قضیه‌ی زیر را به نقل از این مرجع بیان می‌کنیم:

قضیه ۲–۳ سیگنال تمیز $\mathbf{y}_۰ = \mathbf{D}\mathbf{x}_۰$ را در نظر بگیرید. در اینصورت اگر $\|\mathbf{x}_۰\|_۰ = s < \frac{۱}{۲}(۱ + ۱/\mu(\mathbf{D}))$ آنگاه

$$\|\mathbf{x}_۰^\epsilon - \mathbf{x}_۰\|_۲ \leqslant \frac{\epsilon+\delta}{\sqrt{۱-\mu(\mathbf{D})(۲s-۱)}}, \quad \forall \epsilon \geqslant \delta > ۰ \tag{۸–۲}$$

برای مسأله‌ی $P_۱^\epsilon$ هم قضیه‌ی زیر در [۲۰] اثبات شده است:

قضیه ۲–۴ سیگنال تمیز $\mathbf{y}_۰ = \mathbf{D}\mathbf{x}_۰$ را در نظر بگیرید. در اینصورت اگر $\|\mathbf{x}_۰\|_۰ = s < \frac{۱}{۴}(۱ + ۱/\mu(\mathbf{D}))$ آنگاه

$$\|\mathbf{x}_۱^\epsilon - \mathbf{x}_۰\|_۲ \leqslant \frac{\epsilon+\delta}{\sqrt{۱-\mu(\mathbf{D})(۴s-۱)}}, \quad \forall \epsilon \geqslant \delta > ۰ \tag{۹–۲}$$

طبق بحثی که در انتهای بخش ۲–۳ شد، قضیه‌ی ۲–۳ پایداری مسأله‌ی $P_۰^\epsilon$ را فقط برای جواب‌های خیلی تُنُک تضمین می‌کند. در [۳] پایداری این مسأله برای حالت $\|\mathbf{x}_۰\|_۰ < \text{spark}(\mathbf{D})/۲$ (یعنی همان شرط یکتائی تُنُک‌ترین جواب) اثبات شده است. قضیه‌ی زیر را از [۳] بیان می‌کنیم:

قضیه ۲–۵ سیگنال تمیز $\mathbf{y}_۰ = \mathbf{D}\mathbf{x}_۰$ را در نظر بگیرید. در اینصورت اگر $\|\mathbf{x}_۰\|_۰ = s < spark(\mathbf{D})/۲$ آنگاه

$$\|\mathbf{x}_۰^\epsilon - \mathbf{x}_۰\|_۲ \leqslant \frac{\epsilon+\delta}{\sigma_{\min}^{(\ell)}}, \quad \forall \epsilon \geqslant \delta > ۰ \tag{۱۰–۲}$$

که در آن $\ell = ۲s$ و $\sigma_{\min}^{(\ell)}$ برابر است با کوچکتری مقدار تکین[۱] در بین تمامی زیرماتریس‌های متشکل از ℓ ستون از $\mathbf{D}$.

تا این‌جا پُشتوانه‌های تئوری برای یکتائی و پایداری تُنُک‌ترین نمایش یک سیگنال را با بیان قضایائی که در مراجع مربوطه اثبات شده است، ذکر کردیم. در قسمت بعد، تعدادی از الگوریتم‌های عملی را که برای بدست آوردن نمایش تُنُک سیگنال‌ها در سال‌های اخیر پیشنهاد شده است، به اختصار مرور می‌کنیم.

۲-۵ الگوریتم‌های بدست آوردن نمایش تُنُک

طی دهه‌ی اخیر الگوریتم‌های زیادی برای بدست‌آوردن تُنُک‌ترین نمایش یک سیگنال معرفی شده است (به عنوان یک مرور نسبتاً جامع ، [۵۶] و مراجع داخل آن را ببینید). این الگوریتم‌ها را می‌توان به دو دسته‌ی کلّی تقسیم کرد: الگوریتم‌های حریص و الگوریتم‌های مبتنی بر حل مسأله‌ی بهینه‌سازی. الگوریتم‌های حریص به صورت قدم به قدم، یک یا چند اتم را که بیشترین همبستگی با باقی‌مانده‌ی مربوط به نمایش سیگنال دارند انتخاب کرده و با استفاده از آن‌ها این باقی‌مانده را به‌روز می‌کنند. اساس این روش‌ها مبتنی بر تخمین مرحله به مرحله‌ی سیگنال با استفاده از اتم‌های دیکشنری است. دسته‌ی دومِ الگوریتم‌ها هدفشان حل یک مسأله‌ی بهینه‌سازی است که حالت کلّی آن مسأله‌ی (۲-۱) (یا نسخه‌ی پایدار به نویز آن) است. به طور کلّی، الگوریتم‌های حریص نسبت به گروه دیگر الگوریتم‌ها سریع‌تر هستند؛ امّا هزینه‌ی این سرعت بالا، دقّت کمتر این الگوریتم‌ها است. در قسمت بعد، تعدادی از این الگوریتم‌ها را به اختصار مرور می‌کنیم.

۲-۵-۱ الگوریتم‌های حریص

الگوریتم‌های حریص را می‌توان به دو دسته‌ی کلّی تقسیم کرد: دسته‌ی اول در هر گام تنها یک اتم را به عنوان اتمی که بیشترین شباهت را به باقی‌مانده نمایش سیگنال دارد انتخاب می‌کنند. دسته‌ی دوم بیش از یک اتم را انتخاب کرده، سپس طی پیشروی الگوریتم تعدادی از این اتم‌ها را حذف کرده یا اتم‌های جدیدی را به این مجموعه اضافه می‌کنند. الگوریتم‌های MP[۲] و OMP[۳] در دسته‌ی اول، و الگوریتم‌های CoSaMP[۴] و IHT[۵] در دسته‌ی دوم قرار

[۱] Singular value

[۲] Matching Pursuit

[۳] Orthogonal Matching Pursuit

[۴] Compressive Sampling Matching Pursuit

- **هدف:** محاسبه‌ی نمایش تُنُک $\mathbf{y}$ برحسب اتم‌های $\mathbf{D}$
- **مقداردهی اولیه:** $\mathbf{r}^{۰} = \mathbf{y}$ و $\mathbf{x}^{۰} = ۰$
- **شروع الگوریتم:** قرار بده $k = ۱$ و گام‌های زیر را تا رسیدن به شرط توقف انجام بده:
  ۱. محاسبه‌ی همبستگی اتم‌ها با باقی‌مانده: $\mathbf{c}^k = \mathbf{D}^T \mathbf{r}^{k-۱}$
  ۲. انتخاب بهترین اتم: $i^k = \arg\max_i |c_i^k|$
  ۳. به‌روز کردن نمایش تُنُک: $x_{i^k}^k = x_{i^k}^{k-۱} + c_{i^k}^k$
  ۴. به‌روز کردن باقی‌مانده: $\mathbf{r}^k = \mathbf{r}^{k-۱} - c_{i^k}^k \mathbf{d}_{i^k}$
  ۵. چک کردن شرط توقف: اگر شرط توقف برآورده نشده است، قرار بده $k = k + ۱$ و برگرد به گام ۱
- **خروجی:** $\mathbf{x}^k$

شکل ۲-۲: الگوریتم جستجوی تطابق (MP).

دارند.

۲-۵-۱-۱ الگوریتم MP

الگوریتم جستجوی تطابق یا به اختصار MP به عنوان روشی برای تجزیه‌ی اتمی و در [۴۵] معرفی شد. در این الگوریتم، بصورت قدم به قدم اتمی انتخاب می‌شود که بیشترین شباهت (یا به بیان دیگر کم‌ترین فاصله) را با باقی‌مانده‌ی نمایش سیگنال دارد. این باقی‌مانده در قدم اول خود سیگنال بوده و سپس در قدم‌های بعدی از تفاضل سیگنال با تقریب آن بدست می‌آید. معیار توقف این الگوریتم (و سایر الگوریتم‌های حریص) یا رسیدن نُرم باقی‌مانده (یا خطای نمایش) به یک حد از پیش تعیین شده و یا معیار حداکثر تعداد اتم‌ها در نمایش سیگنال است. شبه کُد این الگوریتم در شکل ۲-۲ آورده شده است.

۲-۵-۱-۲ الگوریتم OMP

الگوریتم OMP یکی دیگر از الگوریتم‌های حریص است که در [۵۰] معرفی شده است. اساس این الگوریتم شبیه الگوریتم MP است ولی با یک تفاوت مهم که منجر به بهبود زیاد عملکرد آن می‌شود. در این الگوریتم بر خلاف الگوریتم MP، برای به‌روز کردن نمایش تُنُک، سیگنال روی زیرفضای تولید شده توسط اتم‌های انتخاب شده تا آن مرحله، تصویر می‌شود. به عبارت دیگر، در گام ۳ شکل ۲-۲ داریم $\mathbf{x}^k = \mathbf{D}_{\Gamma^k}^{\dagger}\mathbf{y}$ که در آن Γ^k حاوی اندیس

[۵] Iterative Hard Thresholding

اتم‌هائی است که تا آن گام انتخاب شده‌اند. در گام ۴ نیز داریم $\mathbf{r}^k = \mathbf{r}^{k-1} - \mathbf{D}\mathbf{x}^k$. توجه کنید که این کار حجم محاسبات این الگوریتم را نسبت به الگوریتم MP افزایش می‌دهد؛ ولی عملکرد آن بهبود زیادی پیدا می‌کند.

۲-۵-۱-۳ الگوریتم CoSaMP

در الگوریتم‌های حریص مبتنی بر انتخاب تنها یک اتم در هر گام، مانند دو الگوریتم MP و OMP، اتمی که در هر گام انتخاب می‌شود، به طور قطع تا پایان الگوریتم در نمایش تُنُک سیگنال حضور دارد. به عبارت دیگر، اگر این الگوریتم‌ها در انتخاب یک اتم اشتباه کنند، اثر آن تا پایان الگوریتم وجود داشته و در نتیجه منجر به جواب‌های نادرست می‌شود. احتمال بروز این اشتباه به خصوص زمانی که همبستگی (شباهت) بین اتم‌های دیکشنری زیاد است، بالا می‌رود. این امر به دلیل «حرص زیاد» این دسته از الگوریتم‌ها است. یک راه‌حل برای غلبه بر این مشکل، انتخاب چندین اتم در گامِ انتخاب بهترین اتم‌ها، به عنوان کاندید حضور در نمایش تُنُک سیگنال، و سپس استفاده از تعدادی از این اتم‌ها برای به‌روز کردن نمایش سیگنال است. الگوریتم CoSaMP یکی از الگوریتم‌هائی است که این کار را انجام می‌دهد، که البته پشتوانه‌ی تئوری خوبی هم دارد [۴۷]. این الگوریتم یک نمایش s-تُنُک، یعنی حداکثر با s مؤلفه‌ی غیر صفر، از سیگنال بدست می‌آورد. به عبارت دیگر، مقدار s باید معلوم باشد؛ هر چند همانطور که در [۴۷] پیشنهاد شده است، می‌توان از رابطه‌ی $s \leq n/(2\log(1+m/s))$ به عنوان تخمینی از s استفاده کرد. شبه کُد این الگوریتم در شکل ۲-۳ آورده شده است. در این شکل، منظور از $[.]_s$ ، انتخاب s تا از مؤلفه‌ها با بزرگترین قدرمطلق و صفر کردن بقیه است.

با کمی دقت در شکل ۲-۳ می‌توان فهمید که گام ۵ این الگوریتم مشکل دارد. اینکه برای محاسبه‌ی تخمین نهائی جواب، فقط s مؤلفه‌ی جوابِ حداقل مربعات گام قبلی را انتخاب کرده و بقیه را صفر کنیم از نظر ریاضی دقیق نیست. در واقع اگر مجموعه‌ی اندیس‌های متناظر با این s مؤلفه را با $\hat{T}$ نشان دهیم، گام ۵ این الگوریتم باید بصورت $\mathbf{x}_k = \mathbf{D}_{\hat{T}}^{\dagger}\mathbf{y}$ اصلاح شود. البته این مشکل در الگوریتم جستجوی زیرفضا[1] (SP) [۱۶] حل شده است. تفاوت دیگر این الگوریتم با الگوریتم CoSaMP در گام دوم شکل ۲-۳ است که در آن بجای $2s$ اتم، s اتم انتخاب می‌کند. لازم به توضیح است که دو الگوریتم CoSaMP و SP در یک سال و مستقل از یکدیگر منتشر شده‌اند.

[1] Subspace pursuit

- **هدف:** محاسبه‌ی یک نمایش s-تُنُک از سیگنال $\mathbf{y}$ در دیکشنری $\mathbf{D}$
- **مقداردهی اولیه:** $\mathbf{r}^{۰} = \mathbf{y}$ و $\mathbf{x}^{۰} = ۰$
- **شروع الگوریتم:** قرار بده $k = ۱$ و گام‌های زیر را تا رسیدن به شرط توقف انجام بده:
  ۱. محاسبه‌ی همبستگی اتم‌ها با باقی‌مانده: $\mathbf{c}^k = \mathbf{D}^T\mathbf{r}^{k-۱}$
  ۲. انتخاب $۲s$ اتم با بیشترین شباهت به باقی‌مانده: $\Omega = \text{supp}([\mathbf{c}^k]_{۲s})$
  ۳. ترکیب اندیس اتم‌های جدید با اتم‌های قدیم: $T = \Omega \cup \text{supp}(\mathbf{x}^{k-۱})$
  ۴. تخمین اولّیه از نمایش تُنُک: $\mathbf{x}_T^k = \mathbf{D}_T^{\dagger}\mathbf{y}$ و $\mathbf{x}_{T^c}^k = ۰$
  ۵. تخمین نهائی نمایش تُنُک: $\mathbf{x}^k = [\mathbf{x}^k]_s$
  ۶. به‌روز کردن باقی‌مانده: $\mathbf{r}^k = \mathbf{y} - \mathbf{D}\mathbf{x}^k$
  ۷. چک کردن شرط توقف: اگر شرط توقف برآورده نشده است، قرار بده $k = k + ۱$ و برگرد به گام ۱
- **خروجی:** $\mathbf{x}^k$

شکل ۲-۳: الگوریتم CoSaMP. منظور از $[.]_s$ ، انتخاب s تا از مؤلفه‌ها با بزرگترین قدرمطلق و صفر کردن بقیه است.

۲-۵-۱-۴ الگوریتم IHT

الگوریتم تکراری آستانه‌گذاری سخت یا به اختصار IHT [۸]، یک الگوریتم نسبتاً ساده است که با شروع از یک جواب اولّیه (عموماً بُردار صفر)، سعی می‌کند یک تقریب با s ضریب غیر صفر از نمایش تُنُک سیگنال هدف بدست آورد. اساس این الگوریتم در عبارت زیر خلاصه شده است:

$$\begin{cases} \mathbf{x}_{۰} = ۰ \\ \mathbf{r}_k = \mathbf{y} - \mathbf{D}\mathbf{x}_k \\ \mathbf{x}_{k+۱} = [\mathbf{x}_k + \mathbf{D}^T\mathbf{r}_k]_s, \quad k \geq ۰. \end{cases} \tag{۱۱-۲}$$

دقّت کنید که اگر چه در [۵۶] این الگوریتم جزء الگوریتم‌های حریص در نظر گرفته شده است، امّا به نظر نگارنده، اساس این الگوریتم بیشتر شبیه الگوریتم‌های مبتنی بر حل مسأله‌ی بهینه‌سازی است. توضیح بیش‌تر اینکه، همانطور که در [۸] ذکر شده است، الگوریتم فوق به یک مینیمم محلّی مسأله‌ی زیر همگرا می‌شود:

$$\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{D}\mathbf{x}\|_{۲} \quad \text{subject to} \quad \|\mathbf{x}\|_{۰} \leq s \tag{۱۲-۲}$$

با این توضیح و با توجه به رابطه‌ی (۲-۱۱)، ایده‌ی این الگوریتم را می‌توان استفاده از گرادیان-تصویر[۱] برای حل مسأله‌ی فوق دانست؛ به این ترتیب که بدون در نظر گرفتن قید مسأله، به صورت تکراری تابع هدف را مینیمم کرده و در انتهای هر تکرار، جواب بدست آمده را روی فضای مجاز مسأله تصویر می‌کند؛ یعنی تنها s تا از مؤلفه‌های

[۱] Gradient-Projection

جواب با بزرگترین قدرمطلق را نگه داشته، بقیه را صفر می‌کند.

۲-۵-۱-۵ سایر الگوریتم‌ها

الگوریتم‌های حریص زیادی تاکنون معرفی شده است که در واقع اصول همگی‌شان یکی است ولی در پیچیدگی و دقّت با هم تفاوت دارند. الگوریتم StOMP[۱] [۲۳] الگوریتم دیگری از خانواده‌ی الگوریتم‌های حریص است. این الگوریتم مشابه دو الگوریتم CoSaMP و SP است؛ با این تفاوت که در گام انتخاب اتم، از یک آستانه‌ی از پیش تعیین شده روی ضرب داخلی اتم‌ها با باقی‌مانده، برای انتخاب اتم‌ها استفاده می‌کند. اگر آستانه طوری باشد که در هر مرحله تنها یک اتم انتخاب شود، به الگوریتم OMP می‌رسیم. الگوریتم ISD[۲] که در [۵۹] معرفی شده است در حقیقت تلفیقی از الگوریتم‌های حریص و الگوریتم‌های مبتنی بر حل مسأله‌ی بهینه‌سازی است. این الگوریتم بصورت تکراری بوده، که در هر تکرار دو گام را انجام می‌دهد؛ گام تشخیص اتم‌های فعّال در نمایش سیگنال و گام بازسازی سیگنال، که در این گام یک مسأله‌ی کاهش یافته‌ی نُرمِ یک شبیه (۲-۷) حل می‌شود. برای جزئیات بیشتر درباره‌ی این الگوریتم به [۵۹] مراجعه کنید.

۲-۵-۲ الگوریتم‌های مبتنی بر حل مسأله‌ی بهینه‌سازی

همانطور که قبلاً گفته شد، دسته‌ی دوم الگوریتم‌ها مسأله‌ی بهینه‌سازی (۲-۱) را برای بدست آوردن تُنُک‌ترین نمایش یک سیگنال حل می‌کنند. انتخاب‌های گوناگونی برای تابع تشویق کننده‌ی تُنُک بودن وجود دارد. دسته‌ی کلّی این توابع به صورت $J(\mathbf{x}) = \sum_i \rho(x_i)$ است که در آن $\rho(x)$ تابعی متقارن، یکنوا غیر افزایشی و دارای مشتق یکنوا افزایشی برای $x \geq ۰$ است. چند مثال از این گونه توابع عبارتند از: $\rho(x) = |x|$ ، $\rho(x) = ۱ - \exp(|x|)$ ، $\rho(x) = \log(۱ + |x|)$ و $\rho(x) = |x|/(۱ + |x|)$ [۲۴]. در ادامه چند مورد از الگوریتم‌های معرفی شده تا کنون را به اختصار بررسی می‌کنیم.

۲-۵-۲-۱ خانواده الگوریتم‌های IST

الگوریتم‌های تکراری آستانه‌گذاری-انقباض[۳] (IST) [۱۷، ۲۷] سعی در حل مسأله‌ی غیر مقیّد زیر دارند:

[۱] Stagewise Orthogonal Matching Pursuit

[۲] Iterative Support Detection

[۳] Iterative Shrinkage-Thresholding

$$\min_{\mathbf{x}} f(\mathbf{x}) = \frac{۱}{۲}\|\mathbf{y} - \mathbf{D}\mathbf{x}\|_۲^۲ + \lambda\|\mathbf{x}\|_۱, \qquad (۱۳–۲)$$

که در آن λ پارامتر رگولاریزیشن بوده که نشان‌دهنده‌ی مصالحه‌ی بین میزان تُنُک بودن و خطای نمایش است. به عبارت دیگر، مقادیر بزرگ‌تر این پارامتر منجر به نمایش‌های تُنُک‌تری می‌شود. ایده‌ی کلّی این الگوریتم‌ها تبدیل مسأله‌ی فوق به m مسأله‌ی اسکالر و سپس حل این مسائل است. برای این منظور، جمله‌ی زیر به تابع هدف فوق اضافه می‌شود:

$$d(\mathbf{x}, \mathbf{x}_۰) = \frac{c}{۲}\|\mathbf{x} - \mathbf{x}_۰\|_۲^۲ - \frac{۱}{۲}\|\mathbf{D}\mathbf{x} - \mathbf{D}\mathbf{x}_۰\|_۲^۲ \qquad (۱۴–۲)$$

که در آن $c > \lambda_{\max}(\mathbf{D}^T\mathbf{D})$ یک عدد ثابت است. نقش بُردار $\mathbf{x}_۰$ در ادامه مشخص خواهد شد. در این صورت تابع هدف جدید عبارت است از $\tilde{f}(\mathbf{x}, \mathbf{x}_۰) = f(\mathbf{x}) + d(\mathbf{x}, \mathbf{x}_۰)$.توجه کنید که هدف از اضافه کردن این جمله، تبدیل تابع هدف به گونه‌ای است که بر حسب مؤلفه‌های بُردار $\mathbf{x}$ جداپذیر شود؛ این موضوع به راحتی با ساده کردن تابع $\tilde{f}(\mathbf{x}, \mathbf{x}_۰)$ قابل مشاهده است. به علاوه، همانطور که در ادامه می‌بینیم، این جمله معیاری از نزدیکی دو جواب متوالی در حل تکراری مسأله‌ی (۶–۴۷) است [۱۷]. به عبارت دیگر، $\mathbf{x} \to \mathbf{x}_۰ \;\Rightarrow\; \tilde{f}(\mathbf{x}, \mathbf{x}_۰) \to f(\mathbf{x})$. برای بدست آوردن مینیمم تابع $\tilde{f}(\mathbf{x}, \mathbf{x}_۰)$، مشتق آن را برابر صفر قرار می‌دهیم و معادله‌ی حاصل را برای $\mathbf{x}$ حل می‌کنیم. توجه به این نکته ضروری است که چون تابع قدرمطلق مشتق‌پذیر نیست، باید از زیر–گرادیان[۱] آن استفاده کرد. به این ترتیب به معادله‌ی زیر می‌رسیم:

$$\mathbf{x} - \mathbf{z}_۰ + \frac{\lambda}{c}\mathrm{sgn}(\mathbf{x}) = ۰, \qquad (۱۵–۲)$$

که در آن $\mathbf{z}_۰ = \frac{۱}{c}\mathbf{D}^T(\mathbf{y} - \mathbf{D}\mathbf{x}_۰) + \mathbf{x}_۰$ است. در نهایت، جواب نهائی که مینیمم کننده‌ی سراسری تابع $\tilde{f}(\mathbf{x}, \mathbf{x}_۰)$ است بصورت زیر بدست می‌آید:

$$\mathbf{x}_{opt} = \mathcal{S}_{\frac{\lambda}{c}}(\mathbf{z}_۰), \qquad (۱۶–۲)$$

که در آن $\mathcal{S}_\lambda(.)$ تابع آستانه‌گذاری نَرم[۲] است که مؤلفه به مؤلفه عمل کرده و بصورت زیر تعریف می‌شود:

$$\mathcal{S}_\lambda(a) = \begin{cases} a + \lambda & a \le -\lambda \\ ۰ & |a| \le \lambda \\ a - \lambda & a \ge \lambda \end{cases} . \qquad (۱۷–۲)$$

تا اینجا ما تابع هدف اصلی را به تابعی جدید تبدیل کردیم که برای آن یک مینیمم‌کننده‌ی سراسری به فُرم بسته بدست آمد. حال برای حل مسأله‌ی (۶–۴۷)، تابع $f(\mathbf{x})$ را به صورت تکراری مینیمم می‌کنیم؛ طوری که برای

[۱] Sub-gradient

[۲] Soft-thresholding function

بدست آوردن $\mathbf{x}_{i+1}$ ، تابع $\tilde{f}(\mathbf{x},\mathbf{x}_i)$ را با انتخاب $\mathbf{x}_i = \mathbf{x}_0$ روی $\mathbf{x}$ مینیمم می‌کنیم. به طور خلاصه، تکرارهائی به فُرم زیر انجام می‌شود:

$$\mathbf{x}_{i+1} = \mathcal{S}_{\frac{\lambda}{c}}(\frac{1}{c}\mathbf{D}^T(\mathbf{y}-\mathbf{D}\mathbf{x}_i)+\mathbf{x}_i) \qquad (۲\text{–}۱۸)$$

ایده‌ی فوق، یعنی استفاده از تابع $\tilde{f}(\mathbf{x},\mathbf{x}_0)$ برای بهینه‌کردن تابع $f(\mathbf{x})$، به روش تابع جایگزین[1] یا روش Majorization Minimization(MM) معروف است. برای جزئیات بیشتر درباره‌ی این الگوریتم به [۱۷، ۲۴] مراجعه کنید.

دقت کنید که مانند الگوریتمِ IHT، الگوریتم‌های IST را نیز می‌توان این‌طور تفسیر کرد که برای حل مسأله‌ی (۶–۴۷)، در حقیقت مسأله‌ی زیر را با استفاده از الگوریتم گرادیان–تصویر حل می‌کنند:

$$\min_{\mathbf{x}} f(\mathbf{x}) = \frac{1}{2}\|\mathbf{y}-\mathbf{D}\mathbf{x}\|_2^2 \quad \text{subject to} \quad \|\mathbf{x}\|_1 \le \epsilon(\lambda), \qquad (۲\text{–}۱۹)$$

که در آن، آستانه‌ی $\epsilon(\lambda)$ به نحوی تعیین می‌شود که جواب مسأله‌ی فوق با مسأله‌ی (۶–۴۷) یکسان باشد. در این‌جا، تصویر کردن روی مجموعه‌ی مجاز (یا همان قید مسأله‌ی فوق) منجر به عملیات آستانه‌گذاری نَرم می‌شود.

۲–۵–۲–۲ خانواده الگوریتم‌های IRLS

خانواده الگوریتم‌های IRLS[2] در حالت کلّی سعی در حل مسأله‌ی زیر دارند:

$$\min_{\mathbf{x}} \|\mathbf{x}\|_p^p \quad \text{subject to} \quad \mathbf{y}=\mathbf{D}\mathbf{x}. \qquad (۲\text{–}۲۰)$$

رهیافت این دسته از الگوریتم‌ها برای حل مسأله‌ی فوق، جایگزین کردن تابع هدف (شبه) نُرم ℓ_p با نُرم ℓ_2 وزن‌دهی شده، و انجام تکرارهائی به صورت زیر است:

$$\mathbf{x}^{(k+1)} = \arg\min_{\mathbf{x}} \sum_i w_i^{(k)} x_i^2 = \mathbf{x}^T\mathbf{W}_k\mathbf{x} \quad \text{subject to} \quad \mathbf{y}=\mathbf{D}\mathbf{x}, \qquad (۲\text{–}۲۱)$$

که در آن $\mathbf{W}_k$ یک ماتریس قطری بوده که عناصر روی قطر اصلی با استفاده از جواب تکرار قبلی و به صورت $w_i^{(k)} = |x_i^{(k)}+\sigma|^{p-2}$ محاسبه می‌شود. σ به منظور اجتناب از بروز تقسیم بر صفر به این جمله اضافه شده است و به علاوه، استفاده از یک دنباله‌ی کاهشی برای این کمیت منجر به افزایش توانائی الگوریتم در بازیابی نمایش تُنُک می‌شود (الگوریتم SL0 در قسمت بعد و نیز [۱۳] را ببینید). به این ترتیب، مسأله‌ی غیرمحدّب (۲–۲۰) (البته به استثنای حالت $p=1$) منجر به حل تعدادی مسأله‌ی محدّب (۲–۲۱) می‌شود. برای بدست آوردن جواب تکرار بعدی، یعنی $\mathbf{x}^{(k+1)}$، کافی است مسأله‌ی (۲–۲۱) را با استفاده از ضرایب لاگرانژ حل کنیم. در نهایت جواب فُرم

[1] Surrogate function

[2] Iteratively Re-weighted Least Squares

بسته‌ی زیر بدست می‌آید:

$$\mathbf{x}^{(k+1)} = \mathbf{W}^{\dagger}{}_k \mathbf{D}^T (\mathbf{D}\mathbf{W}^{\dagger}{}_k \mathbf{D}^T)^{-1} \mathbf{y}, \qquad (۲۲-۲)$$

با پیشروی الگوریتم و در صورت همگرائی، تابع هدف مسأله‌ی (۲-۲۱) به تابع نُرم ℓ_p میل می‌کند. مشکل این الگوریتم‌ها نیاز به محاسبه‌ی معکوس ماتریس در رابطه‌ی (۲-۵) است، طوریکه در مسائل با ابعاد خیلی بالا عملاً فاقد کاربرد می‌شوند. برای جزئیات بیشتر درباره این خانواده‌ی الگوریتم‌ها، [۱۳] و مراجع داخل آن را ببینید.

۲-۵-۲-۳ الگوریتم SL0

الگوریتم SL0[۱] به عنوان تابع تشویق کننده‌ی تُنُک بودن، از تابع زیر استفاده می‌کند:

$$J(\mathbf{x};\sigma) = \sum_i (۱ - f_\sigma(x_i)) = \sum_i (۱ - \exp(\frac{x_i^۲}{۲\sigma^۲})) \qquad (۲۳-۲)$$

توجه کنید که این تابع، تقریبی هموار و مشتق‌پذیر از تابع شبه نُرم صفر است؛ به بیان دیگر داریم $\|\mathbf{x}\|_۰ = \lim_{\sigma\to۰} J(\mathbf{x};\sigma)$. با این توضیح، بازای σ های خیلی کوچک تقریب خوبی از شبه نُرم صفر بدست می‌آید که مشتق‌پذیر بوده و براحتی می‌توان مسأله‌ی بهینه‌سازی متناظر را با الگوریتم‌های مبتنی بر گرادیان، مثل الگوریتم SD[۲] حل کرد. مشکلی که وجود دارد این است که تابع $J(\mathbf{x};\sigma)$ در این حالت تعداد زیادی مینیمم محلّی داشته، امکان توقف الگوریتم در این مینیمم‌های عموماً نامطلوب زیاد است. راه‌حلی که در [۴۶] پیشنهاد شده است، استفاده از یک دنباله‌ی کاهشی از مقادیر σ و حل مسائل متناظر به صورت تکراری و بعلاوه، شروع هر مسأله با استفاده از جواب نهائی مسأله‌ی قبلی است. این ایده، که در الگوریتم IRLS نیز به آن اشاره شد، به «عدم تحدّب تدریجی» یا GNC[۳] معروف است [۷]. در این وضعیت اگرچه احتمال توقف الگوریتم در یک مینیمم محلّی نامطلوب کاسته می‌شود[۴]؛ ولی مسأله‌ی کلّی هم‌چنان غیرمحدّب بوده و لذا تضمینی به همگرائی به یک جواب مینیمم سراسری وجود ندارد. با این حال، همانطور که در [۴۶] نشان داده شده است، این الگوریتم سرعت و دقّت نسبتاً بالائی دارد. برای جزئیات بیشتر و نکات مربوط به پیاده‌سازی این الگوریتم، به مرجع مربوطه مراجعه کنید.

---

[۱] Smoothed L0

[۲] Steepest Descent

[۳] Graduated Non-Convexity

[۴] به این دلیل که تعدادی از این مینیمم‌ها (بسته به مقدار σ) از بین رفته و به تعبیری «پُر» می‌شوند.

۲-۵-۲-۴ سایر الگوریتم‌ها

دسته‌ی الگوریتم‌های مبتنی بر حل مسأله‌ی بهینه‌سازی برای یافتن تقریب تُنُک یک سیگنال بسیار گسترده بوده [۲۴، ۵۶] و از حوصله‌ی این پایان‌نامه خارج است. به عنوان مثال، هر کدام از خانواده الگوریتم‌های IST و IRLS خود شامل روش‌های گوناگونی است که در سرعت و دقّتِ جواب نهائی با هم رقابت می‌کنند. به علاوه، دسته‌ی دیگری از الگوریتم‌ها وجود دارد که مبتنی بر روش‌های گرادیان بوده، خود شامل چندین روش است. [۵۶] مرجع خوبی برای مروری مختصر بر این الگوریتم‌ها است.

۲-۶ مقدمه‌ای بر حسگری فشرده

حسگری فشرده[۱] روشی نوین برای ضبط و فشرده‌سازی داده‌ها است که نخستین بار در [۹، ۱۸] معرفی شد. روش‌های کلاسیک نمونه‌برداری مبتنی بر نظریه‌ی شانون-نایکوئیست[۲] است، که بیان می‌کند برای بازسازی کامل یک سیگنال، نرخ نمونه‌برداری باید حداقل به اندازه‌ی دو برابر بزرگترین فرکانس موجود در آن باشد. اساس حسگری فشرده بر این نکته استوار است که بسیاری از سیگنال‌ها از جمله تصویر، صوت و سیگنال‌های زلزله در یک پایه‌ی معین (یا یک دیکشنری)، تُنُک یا فشرده‌پذیر[۳] هستند. بر این اساس، حسگری فشرده بیان می‌کند که نرخ اطلاعاتِ (میزان اطلاعات موجود در واحد زمانِ) یک سیگنال پیوسته می‌تواند خیلی کمتر از پهنای باند آن باشد، یا تعداد درجات آزادی یک سیگنال گسسته خیلی کمتر از طولِ (محدود) آن است.

ابزارهای متداول ضبط (و انتقال) داده، مانند دوربین عکاسی، به این ترتیب عمل می‌کنند که ابتدا حجم زیادی از داده (طبق تئوری شانون-نایکوئیست) را ضبط کرده، سپس به کمک یک روش فشرده سازی، مانند روش‌های کُدینگ حوزه‌ی تبدیل[۴]، حجم داده را کاهش می‌دهند. به عبارت بهتر، ابتدا داده‌ای با طول زیاد ضبط می‌شود، سپس به حوزه‌ای دیگر (مثلاً با استفاده از تبدیل ویولت) منتقل شده، در آنجا روی ضرایب آستانه گذاری انجام می‌شود، و نهایتاً ضرایب باقی‌مانده به همراه موقعیت آن‌ها کُد می‌شوند. این روش امّا به دلایل زیر ناکارآمد است [۴]. اولاً، تعداد نمونه‌های اولیه‌ی سیگنال که ضبط می‌شود در بسیاری از کاربردها، مانند تصویربرداری،

[۱] Compressed Sensing

[۲] Shannon-Nyquist

[۳] به این معنی که دامنه‌ی ضرایب آن‌ها در آن پایه به طور نمائی کاهش می‌یابد.

[۴] Transform Coding

بسیار زیاد است که خود منجر به هزینه‌ی زیاد به دلیل استفاده از حسگرهای ظریف‌تر می‌شود. دوماً، باید همه‌ی ضرایب تبدیل برای کُد کردن محاسبه شوند؛ اگرچه نهایتاً تنها تعداد کمی از آن‌ها نگه داشته می‌شوند، و سوماً، علاوه بر مقدار ضرایب، موقعیت آن‌ها نیز باید کُد شود. حسگری فشرده برای غلبه بر این مشکلات، یک سیستم اندازه‌گیری (یا ضبطِ داده) معرفی می‌کند که همزمان عمل فشرده‌سازی را نیز انجام می‌دهد؛ به عبارت دیگر، تنها «اطلاعات مفید» موجود در سیگنال را اندازه می‌گیرد.

سیگنال $\mathbf{x} \in \mathbb{R}^m$ را در نظر بگیرید که در پایه‌ی $\mathbf{\Psi} = \{\psi_i\}_{i=1}^m$ دارای نمایش s-تُنُک $\mathbf{s}$ است؛ یعنی $\mathbf{x} = \mathbf{\Psi s}$، و به علاوه، تنها $s \ll m$ ضریبِ $\mathbf{s}$ غیر صفر است. حسگری فشرده برای ضبط سیگنال $\mathbf{x}$، تعداد $n \ll m$ اندازه‌گیری از این سیگنال را ضبط می‌کند. هر اندازه‌گیری با استفاده از ضرب داخلی سیگنال با یک بُردار مشخص از ستون‌های $\mathbf{\Phi} = \{\phi_i\}_{i=1}^n$ بدست می‌آید، طوریکه برای اندازه‌گیری i اُم داریم $y_i = \langle \mathbf{x}, \phi_i \rangle$. این عملیات به فُرم ماتریسی زیر است:

$$\mathbf{y} = \mathbf{\Phi x} = \mathbf{\Phi \Psi s} = \mathbf{\Theta s}. \qquad (۲۴-۲)$$

ماتریس اندازه‌گیری $\mathbf{\Phi}$ وفقی نیست؛ یعنی مستقل از سیگنال $\mathbf{x}$ است. این ماتریس باید به گونه‌ای طراحی شود که قادر باشد اطلاعات مفید سیگنال را ضبط کند و به علاوه، به عنوان یکی از ملزومات حسگری فشرده، باید نسبت به ماتریس $\mathbf{\Psi}$ ناهم‌بسته باشد. به بیان دیگر، حداکثر ضریب همبستگی ستون‌های دو ماتریس $\mathbf{\Phi}$ و $\mathbf{\Psi}$ باید به اندازه‌ی کافی کوچک باشد. یک دسته از این ماتریس‌ها، که با دقّت خوبی شرایط بازسازی سیگنال را فراهم می‌کنند، ماتریس‌های تصادفی است. به عبارت دیگر، ما برای ضبط سیگنال یک سری اندازه‌گیری تصادفی از آن را ضبط می‌کنیم. در نهایت، برای بازسازی سیگنال از روی اندازه‌گیری‌های آن، لازم است یک مسأله‌ی بازسازی نمایش تُنُک حل شود.

در بسیاری از کاربردها از جمله تصویربرداری پزشکی، به خصوص روش MRI[1]، یا مبدل‌های پُرسرعت آنالوگ به دیجیتال، مقدار داده‌ی اندازه‌گیری شده از اهمیت زیادی برخوردار است؛ به عبارت دیگر، برای داشتن یک کیفیت مشخص، هر چه به اندازه‌گیری‌های کم‌تری نیاز داشته باشیم، سرعت پردازش بالارفته و به تناسب آن، هزینه نیز کم‌تر می‌شود. در چنین کاربردهائی، حسگری فشرده نقشی اساسی ایفا می‌کند. به عنوان کاربرد حسگری فشرده در MRI، مرجع [۴۳] را ببینید.

[1] Magnetic Resonance Imaging

۲-۷ جمع‌بندی

در این فصل مباحث تئوری نمایش تُنُک را به همراه تعدادی از الگوریتم‌های موجود برای بازیابی این نمایش، به اختصار مرور کردیم. گفتیم که هسته‌ی مرکزی بحث نمایش تُنُک، یک دستگاه معادلات خطی فرومعیّن است و با بیان چند قضیه، که طی چند سال گذشته اثبات شده است، دیدیم که جواب‌های به اندازه‌ی کافی تُنُک این دستگاه معادلات یکتا بوده و به علاوه، مسائل معرفی شده برای بازیابی نمایش تُنُک (با استفاده از شبه نُرم صفر و نُرم یک) نسبت به یک نویز کوچک پایدارند. تعدادی از الگوریتم‌های موجود را، که در حالت کلّی شامل دسته‌ی الگوریتم‌های حریص و الگوریتم‌های مبتنی بر حل یک مسأله‌ی بهینه‌سازی است، به اختصار مرور کردیم. در حالّت کلّی، سرعت الگوریتم‌های حریص نسبت به دسته‌ی دوم الگوریتم‌ها بیشتر بوده، ولی دقّت آن‌ها کم‌تر است. در انتها، حسگری فشرده را به طور مختصر معرفی کردیم. دیدیم که طبق این رویکرد، می‌توان عملیات ضبط و فشرده‌سازی داده‌ها را به طور همزمان انجام داد. هزینه‌ی این کار امّا نیاز به حلّ یک مسأله‌ی بازیابی نمایش تُنُک است.

در فصل بعد، بحث «آموزش دیکشنری» را برای پردازش تُنُک سیگنال‌ها معرفی می‌کنیم. یک دیکشنری «مناسب» را تعریف خواهیم کرد و نقش آن را در بازیابی نمایش تُنُک بیان می‌کنیم. هم‌چنین، تعدادی از الگوریتم‌هائی را که تاکنون معرفی شده‌اند به اختصار مرور خواهیم کرد.

فصل ۳

آموزش دیکشنری

۳-۱ مقدمه

در فصل قبل مباحث پایه‌ای نمایش تُنُک را به همراه تعدادی از الگوریتم‌های موجود به طور مختصر مرور کردیم. دیدیم که هدف ما تقریب زدن یک سیگنال به صورت یک ترکیب خطی متشکّل از کم‌ترین تعداد اتم‌ها از یک دیکشنری فوق کاملِ[۱] مشخص است. فرض ما تا این‌جا بر این بود که این دیکشنری کاملاً معلوم است. امّا لازم است که برای هر کاربردی مانند فشرده‌سازی، نویززدائی، تشخیص الگو[۲] و ... یک دیکشنری مناسب، یا اصطلاحاً تُنُک کننده[۳]، داشته باشیم. تمرکز این فصل روی طراحی دیکشنری برای یک کلاس مشخص از داده‌ها است. ابتدا مروری مختصر خواهیم داشت بر تاریخچه‌ی دیکشنری‌ها یا همان تبدیل‌ها[۴]. سپس مسأله‌ی آموزش دیکشنری را معرفی می‌کنیم. در ادامه، یکتائی دیکشنری و تعدادی از الگوریتم‌های معرفی شده برای آموزش دیکشنری را بررسی خواهیم کرد.

---

[۱] Overcomplete

[۲] Pattern recognition

[۳] Sparsifying

[۴] در ادامه‌ی این پایان‌نامه، دو کلمه‌ی «دیکشنری» و «تبدیل» را به جای هم و بعنوان دو کلمه‌ی معادل به کار می‌بریم؛ اگرچه اصطلاح «دیکشنری» نخستین بار در سال ۱۹۹۳ و در [۴۵] استفاده شد.

۳-۲ مروری مختصر بر تاریخچه‌ی دیکشنری‌ها

نمایش سیگنال در یک پایه‌ی مناسب از دیرباز مورد توجه بوده است. برای این منظور، سیگنال بر حسب تعدادی سیگنال پایه یا اتم بسط داده می‌شود. هر کدام از این اتم‌ها در حقیقت یک ویژگی از سیگنال را توصیف می‌کنند. در بسیاری از کاربردها لازم است که «ساده‌ترین نمایش» را برای یک سیگنال داشته باشیم. همانطور که در فصل قبل گفتیم، بسیاری از سیگنال‌های (گسسته‌ی) طبیعی اطّلاعاتی به مراتب کمتر از طولشان دارند. به عبارت دیگر، «بُعد ذاتی» آن‌ها از «بُعد ظاهری» آن‌ها خیلی کوچکتر است. بنابراین، اتم‌های مورد استفاده برای نمایش یک سیگنال باید ویژگی‌های مهم و به طور کلّی اطّلاعات مفید آن را استخراج کنند. بعنوان مثال، در استخراج ویژگی[۱] برای تشخیص الگو، نمایش سیگنال باید ویژگی‌های برجسته‌ی آن را توصیف کند.

در ساده‌ترین حالت، یک دیکشنری عبارت است از یک پایه‌ی متعامد. در این حالت، ضرایب بسط یا نمایش سیگنال به سادگی با استفاده از ضرب داخلی آن با اتم‌ها بدست می‌آید. در کلّی‌ترین حالتِ یک «پایه»، که شامل اتم‌های مستقل خطی ولی نه لزوماً متعامد است، ضرایب نمایش سیگنال از ضرب داخلی آن با اتم‌های دوگان[۲] بدست می‌آید. از معروف‌ترینِ این تبدیل‌ها می‌توان به تبدیل فوریه[۳] اشاره کرد. در حالت گسسته، که موسوم به DFT[۴] است، اتم‌های این تبدیل عبارتند از $\{\mathbf{d}_k \in \mathbb{R}^n : \mathbf{d}_k(\ell) = \exp(-j۲\pi k(\ell-۱)/n),\ \ell = ۱, ۲, ..., n\}_{k=۱}^{n}$. هر اتم عبارت است از یک سینوسی مختلط با یک فرکانس منحصر به فرد و بعلاوه، اتم‌های متمایز بر هم عمودند. به این ترتیب، این اتم‌ها محتوای فرکانسی یک سیگنال را توصیف می‌کنند. با وجود محاسبات سریعِ بسط سیگنال (الگوریتم FFT[۵])، این تبدیل امّا تنها برای دسته‌ی محدودی از سیگنال‌ها (یعنی سیگنال‌های هموار یا متمرکز[۶] در حوزه‌ی فرکانس) مناسب است. این مشکل فقط اختصاص به تبدیل فوریه ندارد، بلکه تمامی تبدیل‌های به اصطلاح «کامل»[۷]، یعنی تبدیل‌هائی که در آن تعداد اتم‌ها با بُعد سیگنال برابر است، این مشکل را دارند؛ یعنی تنها برای سیگنال‌هائی با ساختار ساده مناسب‌اند. به عبارت دیگر، این تبدیل‌ها تنها بخشی از ویژگی‌های یک سیگنال

[۱] Feature Extraction
[۲] Dual
[۳] Fourier transform
[۴] Discrete Fourier Transform
[۵] Fast Fourier Transform
[۶] Localized
[۷] Complete

را، که در حالت کلّی دارای ساختار پیچیده‌ای است، می‌توانند توصیف کنند. به عنوان یک مثال شهودی از این موضوع، یک دیکشنری از لغات را تصور کنید. اگر کلمه‌ی «اتم» (یا لااقل کلمه‌های هم‌خانواده با آن) در این دیکشنری وجود نداشته باشد، ما برای توصیف آن لازم است از تعداد زیادی از لغات این دیکشنری استفاده کنیم؛ که این خود منجر به پیچیدگی جمله می‌شود. بنابراین، هر چه این دیکشنری «غنی‌تر» باشد، ما قادر به توصیف پدیده‌های بیشتری و به ساده‌ترین صورت ممکن خواهیم بود.

برای غلبه بر این مشکل، دیکشنری‌های «فوقِ کامل» معرفی شدند. این تبدیل‌ها در جامعه‌ی پردازش سیگنال و خصوصاً آنالیز هارمونیک[۱] به «فریم»[۲] معروف‌اند [۳۹]. در این دیکشنری‌ها، تعداد اتم‌ها از بُعد سیگنال بیشتر است و بنابراین قادراند ویژگی‌های بیشتری از سیگنال را توصیف کنند. واضح است که در این وضعیت، اتم‌ها وابسته‌ی خطی‌اند. به عنوان یک تعریف دقیق‌تر، گوئیم خانواده‌ی اتم‌های $\mathbf{D} = \{\mathbf{d}_i\}_{i=1}^m$ تشکیل یک فریم برای فضای $\mathbb{R}^n$ می‌دهند، هر گاه دو عدد ثابت $0 < A \leq B$ وجود داشته باشند بطوری که:

$$\forall \mathbf{y} \in \mathbb{R}^n : \ A\|\mathbf{y}\|_2^2 \leq \sum_{i=1}^m |\langle \mathbf{d}_i, \mathbf{y} \rangle|^2 \leq B\|\mathbf{y}\|_2^2. \qquad (۱-۳)$$

همانطور که پایه‌های متعامد (یا متعامدِ یکّه؛ که علاوه بر تعامد اتم‌ها، نُرم همگی برابر یک است) ساده‌ترین محاسبات را در بین تبدیل‌های کامل دارند، در بین فریم‌ها نیز چنین تبدیل‌هائی به نام Tight Frame وجود دارد که ساده‌ترین محاسبات را دارند. برای این دسته از فریم‌ها داریم $A = B$. به عبارت دیگر، $\sum_{i=1}^m |\langle \mathbf{d}_i, \mathbf{y} \rangle|^2 = A\|\mathbf{y}\|_2^2$. در نتیجه، ضرایب نمایش یک سیگنال در یک Tight Frame به سادگی از ضرب داخلی سیگنال با هر یک از اتم‌ها بدست می‌آید؛ به عبارت دیگر $\mathbf{y} = \frac{1}{A} \sum_{i=1}^m \langle \mathbf{d}_i, \mathbf{y} \rangle \mathbf{d}_i$. به عنوان مثالی از یک فریم می‌توان به تبدیل DCT فوق کامل اشاره کرد.

تبدیل‌هائی که تا این‌جا بررسی کردیم، ثابت بوده و به عبارتی وابسته به سیگنال نیستند. به این تبدیل‌ها، «دیکشنری‌های از پیش تعریف شده»[۳] یا «دیکشنری‌های غیروفقی»[۴] می‌گویند. اگرچه فریم‌ها نسبت به تبدیل‌های کامل قابلیّت بیش‌تری در نمایش تُنُک سیگنال‌ها دارند؛ امّا با این وجود، قابلیّت وفق کردن به محتویات سیگنال‌های تحت بررسی را ندارند. به عنوان مثال، اگرچه یک تبدیل DCT فوق کامل نسبت به یک تبدیل DCT کامل نمایش تُنُک‌تری برای یک تصویر بدست می‌دهد؛ امّا برای بسیاری از تصاویر نمی‌تواند نمایش به اندازه‌ی کافی تُنُکی

[۱] Harmonic Analysis

[۲] Frame

[۳] Predefined Dictionaries

[۴] Non-Adaptive Dictionaries

داشته باشد. بعنوان یک نمونه، این تبدیل برای تصاویری که شامل فقط بافت[۱] هستند، نمایش تُنُکی دارد، ولی برای تصاویر دارای ساختارهای پیچیده‌تر مانند انواع لبه‌ها، نمایش مناسبی ندارد[۲] [۵۱]. یک راه برای بدست آوردن دیکشنری‌های فوق کاملِ ثابتِ کارآتر، ترکیب پایه‌های مختلف است. هر پایه یک یا چند ویژگی از سیگنال را توصیف می‌کند. بعنوان مثال، برای سیگنال‌های همواری که شامل تعدادی نقطه‌ی یکّه یا تکینی[۳] هستند، می‌توانیم دیکشنری‌های DFT و دلتای دیراک را (که همان ماتریس همانی است) با هم ترکیب کنیم. این ایده نخستین بار در [۱۴] مطرح شد[۴].

اولّین تلاش در جهت بدست آوردن دیکشنری‌های وفقی مربوط به تبدیل‌های کامل است. در این زمینه، تبدیل KLT[۵] که به تجزیه به مؤلّفه‌های اساسی یا PCA[۶] نیز معروف است، معرفی شد [۳۸]. این تبدیل خطی بوده و به دسته‌ای از سیگنال‌ها با یک توزیع آماری معلوم وفق داده می‌شود. در این تبدیل، یک زیرفضای با بُعد کم به داده‌ها برازش می‌شود. به بیان دقیق‌تر، فرض کنید ماتریس کواریانس داده‌ها برابر با $\mathbf{\Sigma}$ است. این ماتریس یا معلوم بوده و یا به صورت $\mathbf{\Sigma} \approx \frac{1}{L}\mathbf{Y}\mathbf{Y}^T$ از روی داده‌های $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_L]$ که $\mathbf{y}_i \in \mathbb{R}^n$ تخمین زده می‌شود. اگر تجزیه به مقادیر ویژه‌ی[۷] این ماتریس به صورت $\mathbf{\Sigma} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ باشد، در اینصورت، اتم‌های دیکشنری KLT عبارتند از ستون‌های ماتریس $\mathbf{U}$، که در حقیقت همان مؤلّفه‌های اساسی یا بُردارهای ویژه‌ی ماتریس کواریانس هستند. با وجود اینکه این تبدیل بهترین زیرفضا (به مفهوم حداقل کردن نُرم خطا) را به داده‌ها برازش می‌کند؛ امّا به حجم محاسبات زیادی به خصوص در ابعاد بالا، نیاز دارد. نشان داده شده است که تبدیل DCT کامل، به عنوان یک تبدیل ثابت، تقریب خوبی از تبدیل KLT برای تصاویر طبیعی است [۳۲].

از دیدگاه آماری، فرض ضمنی تبدیل KLT این است که داده‌ها توزیعی گوسی دارند؛ بنابراین این تبدیل بیش‌تر برای چنین داده‌هائی مناسب است و بعلاوه، در بسیاری از موارد، داده‌ها ذاتاً نه در تنها یک زیرفضا، بلکه در چندین زیرفضا با ابعاد مختلف قرار دارند. این در حالی است که تبدیل KLT تنها «یک» زیرفضا را به داده‌ها

---

[۱] Texture

[۲] لازم به ذکر است که تاکنون تبدیل‌های ثابت مختلفی برای نمایش مناسب‌تر تصاویر معرفی شده است که از آن جمله می‌توان به تبدیل‌های Curvelet، Contourlet، Ridgelet، Bandelet و ... اشاره کرد. برای اطلاّعات بیش‌تر، [۵۱] و مراجع داخل آن را ببینید.

[۳] Singularity

[۴] لازم به ذکر است که گزارش فنّی این مقاله در سال ۱۹۹۵ منتشر شده است.

[۵] Karhunen-Loève Transform

[۶] Principal Component Analysis

[۷] Eigenvalue Decomposition

برازش می‌کند. به عنوان تعمیمی از PCA، الگوریتم PCA تعمیم یافته یا به اختصار GPCA[۱] معرفی شده است [۵۸]. این الگوریتم با رویکردی جبری، چندین زیرفضا با بُعد کم را به داده‌ها برازش می‌کند؛ طوریکه هر داده تنها در یکی از این زیرفضاها قرار دارد. به عبارت دیگر، اتم‌های زیرفضاهای مختلف نمی‌توانند با هم در نمایش یک سیگنال نقش داشته باشند. به این ترتیب، GPCA همان نقش PCA را منتها برای بدست آوردن دیکشنری‌های فوق کامل دارد.

ذکر این نکته ضروری است که یک دسته از الگوریتم‌ها برای آموزش دیکشنری، مشابه الگوریتم GPCA مبتنی بر خوشه‌بندی زیرفضا[۲] هستند (به عنوان مثال [۳۳]). بعبارت دیگر، فرض این الگوریتم‌ها بر این است که داده‌ها در چندین زیرفضا قرار دارند؛ سپس با خوشه‌بندی داده‌های آموزشی، این زیرفضاها را تخمین می‌زنند. امّا اگر داده‌ها واقعاً چنین ساختاری نداشته باشند یا لااقل تعداد داده‌های آموزشی و کیفیت آن‌ها[۳] مناسب نباشد، نمی‌توان از این الگوریتم‌ها انتظار نتیجه‌ای رضایت‌بخش داشت. بعنوان مثال، بلوک‌هائی[۴] از یک تصویر طبیعی را اگر به صورت بُردار نشان دهیم، این بُردارها در حالت کلّی روی یک «مَنیفُلد»[۵] قرار دارند نه یک زیرفضای خطی. توضیح اینکه، یک مَنیفُلد ساختاری هندسی است که به طور محلّی نشان‌دهنده‌ی یک زیرفضای اقلیدسی است (بعنوان مثال، یک رویه‌ی دو بُعدیِ دارای انحنا در فضای سه بُعدی، یک مَنیفُلد است). بنابراین، لازم است بعنوان تقریب مَنیفُلد، از تعداد زیادی زیرفضای خطی استفاده کنیم و در نهایت، برای بدست آوردن دیکشنری نهائی، این زیرفضاها را هرَس[۶] کنیم؛ بعبارت دیگر، اتم‌های مشابه را حذف کرده و به نحوی این زیرفضاها را با هم تلفیق[۷] کنیم. در کاربردهائی مثل پردازش تصویر، این کار می‌تواند منجر به حجم محاسبات زیاد (برای بدست آوردن تعداد زیادی زیرفضا و ترکیب آن‌ها) شود. چنین الگوریتم‌هائی بیشتر برای کاربردهائی مانند جداسازی کور منابع[۸] (BSS)، برای تخمین ماتریس مخلوط‌کننده [۳۳]، و نیز خوشه‌بندی چهره[۹] و قطعه‌بندی[۱۰] تصویر و حرکت، مناسب هستند [۵۸].

---

[۱] Generalized PCA

[۲] Subspace Clustering

[۳] به این معنی که آیا برای توصیف یک زیرفضا به اندازه‌ی کافی سیگنال از آن موجود هست یا خیر.

[۴] Block

[۵] Manifold

[۶] Prune

[۷] Merge

[۸] Blind Source Separation

[۹] Face Clustering

[۱۰] Segmentation

۳-۳ آموزش دیکشنری

همانطور که در بخش قبل دیدیم، دیکشنری‌های ثابت یا از پیش طراحی شده اگرچه محاسبات سریعی دارند؛ امّا عموماً (بسته به کلاس سیگنال‌های تحت بررسی) نمی‌توانند نمایشی به اندازه‌ی کافی تُنُک ارائه دهند. برای غلبه بر این مشکل، بحث «آموزش دیکشنری» طی دهه‌ی اخیر مورد توجّه قرار گرفته است. در این رهیافت، ابتدا تعدادی «داده‌ی آموزشی»[۱] شبیه به سیگنال‌های تحت بررسی انتخاب می‌شود. سپس طی یک الگوریتمِ آموزش، اتم‌های دیکشنری طوری بهینه می‌شوند که تا حد امکان، نمایان‌گر برجسته‌ترین ویژگی‌های (مشترک) داده‌های آموزشی بوده و درنتیجه، نمایشی به اندازه‌ی کافی تُنُک برای سیگنال‌های آموزشی ارائه دهند.

رابطه‌ای نزدیک بین آموزش دیکشنری برای نمایش تُنُک و کوانتیزاسیون بُرداری[۲] [۳۱] وجود دارد. در کوانتیزاسیون بُرداری، هدف طراحی یک کتاب کُد[۳] شامل m بُردار کُد[۴]، $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_m]$، به گونه‌ای است که هر یک از داده‌های آموزشی، یعنی ستون‌های ماتریس $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_L]$ که $\mathbf{y}_i \in \mathbb{R}^n$، برحسب دقیقاً یکی از بُردارهای کُد، که نزدیک‌ترین فاصله را (عموماً بر مبنای نُرم ℓ_2) با آن دارد، قابل نمایش باشد. این نمایش در حقیقت حالت حدّی نمایش تُنُک است. بعبارت دیگر، در حالی که در نمایش تُنُک هر داده می‌تواند از بیش از یک اتم برای توصیف خود استفاده کند، در کوانتیزاسیون بُرداری، هر داده تنها از یک بُردار کُد (در اینجا همان اتم) استفاده می‌کند و بعلاوه، ضریب نمایش نیز برابر با ۱ است. در نتیجه، در این وضعیت تُنُک‌ترین نمایش ممکن را خواهیم داشت. برای یک کتاب کُد معلومِ $\mathbf{C}$، هر کدام از داده‌ها با بُردار کُدی نمایش داده می‌شود که بیش‌ترین شباهت را نسبت به بقیه‌ی اتم‌ها با آن دارد. در اینصورت اگر بُردار کُد متناظر با نمونه‌ی $\mathbf{y}_i$ برابر با $\mathbf{c}_j$ باشد، می‌توان نوشت $\mathbf{y}_i = \mathbf{C}\mathbf{x}_i$، که در آن $\mathbf{x}_i = \mathbf{e}_j$ و $\mathbf{e}_j$ ستون j اُم ماتریس همانیِ $\mathbf{I}_{m\times m}$ است. اگر خطای این نمایش را بصورت $e_i = \|\mathbf{y}_i - \mathbf{C}\mathbf{x}_i\|_2^2$ نشان دهیم، خطای کلّی برابر است با:

$$E = \sum_{i=1}^{L} e_i = \|\mathbf{Y} - \mathbf{C}\mathbf{X}\|_F^2. \qquad (۲-۳)$$

به این ترتیب، هدف کوانتیزاسیون بُرداری یافتن یک کتاب کُد به گونه‌ای است که خطای فوق مینیمم شود. به بیان دقیق‌تر، مسأله‌ی زیر حل می‌شود:

---

[۱] Training data

[۲] Vector Quantization

[۳] Code Book

[۴] Code Vector

$$\min_{\mathbf{C},\mathbf{X}} \|\mathbf{Y}-\mathbf{C}\mathbf{X}\|_F \quad \text{subject to} \quad \mathbf{x}_i = \mathbf{e}_j,\ i = ۱,\ldots,L. \tag{۳–۳}$$

با توجّه به اینکه مسأله‌ی فوق نسبت به دو متغیّر $\mathbf{C}$ و $\mathbf{X}$ توأماً محدّب نیست، بنابراین در حالت کلّی تعداد زیادی مینیمم محلّی دارد. عمومی‌ترین الگوریتم برای طراحی کتاب کُد، الگوریتم K-means است. این الگوریتم با شروع از یک کتاب کُد اولیه، دو گام را به صورت تکراری انجام می‌دهد. گام نخست، «خوشه‌بندی»[۱] یا «کُدینگ تُنُک»[۲] است. در این گام، با استفاده از کتاب کُد فعلی، هر داده به بُردار کُد مناسب خود تخصیص داده می‌شود. در نتیجه تعدادی خوشه متناظر با هر بُردار کُد بدست می‌آید. گام دوم، «به روز کردن بُردارهای کُد» است. در این گام، هر بُردار کُد با استفاده از میانگین داده‌های موجود در خوشه‌ی خود به روز می‌شود[۳]. این دو گام تا رسیدن به یک خطای قابل قبول تکرار می‌شود. تضمینی وجود ندارد که این الگوریتم به مینیمم کننده‌ی سراسری[۴] مسأله‌ی (۳–۳) همگرا شود، ولی بعد از هر تکرار، خطای کلّی یا کم می‌شود و یا ثابت می‌ماند [۳۱].

با توضیحات فوق، طبیعی به نظر می‌رسد که آموزش دیکشنری برای نمایش تُنُک در حقیقت تعمیمی از کوانتیزاسیون بُرداری و الگوریتم‌های آموزش دیکشنری هم تعمیمی از الگوریتم K-means است[۵]. بعبارت دیگر، اولاً قید سنگین وجود تنها یک اتم در نمایش سیگنال‌ها، تبدیل به «نمایش تُنُک» می‌شود (یعنی استفاده‌ی از کم‌ترین تعداد اتم‌ها؛ نه الزاماً یکی). در نتیجه، مسأله‌ی (۳–۳) به مسأله‌ی کلّی‌تر زیر تبدیل می‌شود:

$$\min_{\mathbf{D},\mathbf{X}} \|\mathbf{Y}-\mathbf{D}\mathbf{X}\|_F \quad \text{subject to} \quad \|\mathbf{x}_i\|_۰ \le T_۰,\ i = ۱,\ldots,L, \tag{۴–۳}$$

که $T_۰$ برابر است با حداکثر تعداد مجاز اتم‌ها برای نمایش هر سیگنال آموزشی. دوماً، اکثر الگوریتم‌های آموزش دیکشنری مبتنی بر ماهیت تکراری الگوریتم K-means و دو گام موجود در آن هستند. در گام اول، نمایش تُنُک سیگنال‌ها در دیکشنری فعلی محاسبه می‌شود. در گام دوم، دیکشنری طوری به‌روز می‌شود که خطای کلّی نمایش تُنُک سیگنال‌ها در گام اول، یا مینیمم شود یا لااقل کاهش یابد. الگوریتم‌های مختلفی که طی سال‌های گذشته برای آموزش دیکشنری معرفی شده‌اند عموماً در نحوه‌ی انجام این دو گام اختلاف دارند.

---

[۱] Clustering

[۲] Sparse Coding

[۳] وجه تسمیه‌ی این الگوریتم همین‌جا است که در آن به تعداد بُردارهای کُد (که به جای K ما این تعداد را با m نشان داده‌ایم) میانگین محاسبه می‌شود.

[۴] Global Minimum

[۵] لازم به توضیح است که این ادّعا در مورد همه‌ی الگوریتم‌های آموزش دیکشنری صدق نمی‌کند. در واقع همانطور که در قسمت قبل گفتیم، تعدادی از الگوریتم‌ها مبتنی بر خوشه‌بندی زیرفضا هستند.

۳-۴ یکتائی دیکشنری

به مسأله‌ی آموزش دیکشنری می‌توان به دید فاکتوریزه کردن ماتریس[1] نگاه کرد [۳۴، ۲]. با این دید، ما یک ماتریس $\mathbf{Y} \in \mathbb{R}^{n \times L}$ شامل L سیگنال که $L \gg n$ داریم و هدف فاکتوریزه کردن این ماتریس به صورت $\mathbf{Y} = \mathbf{DX}$ است که $\mathbf{D} \in \mathbb{R}^{n \times m}$ ، $n < m \ll L$ و $\mathbf{X} \in \mathbb{R}^{m \times L}$ یک ماتریس تُنُک است که هر ستون آن حداکثر s درایه‌ی غیر صفر دارد. یک سوال اساسی که در این‌جا مطرح می‌شود این است که آیا اصولاً این فاکتوریزیشن یکتا است؟ مقاله‌ی [۲] در سال ۲۰۰۶ برای نخستین بار این مسأله را بررسی کرده و با اثبات قضیه‌ی زیر نشان داده است که تحت شرایطی پاسخ به سؤال فوق مثبت است:

قضیه ۳-۱ تحت فرض‌های زیر، فاکتوریزه کردن ماتریس $\mathbf{Y}$ به صورت $\mathbf{Y} = \mathbf{DX}$، که ستون‌های ماتریس $\mathbf{D}$ نُرم واحد داشته و هر ستون ماتریس $\mathbf{X}$ حداکثر s درایه‌ی غیر صفر دارد، با در نظر گرفتن جایگشت ستون‌ها و علامت، یکتا است:

۱. $\forall i : \ \|\mathbf{x}_i\|_0 < \frac{spark(\mathbf{D})}{2}$

۲. برای هر ترکیب خطّی ممکن از s ستون از ماتریس $\mathbf{D}$، حداقل به تعداد $s+1$ سیگنال (ستون) در ماتریس $\mathbf{Y}$ وجود دارد. در نتیجه، $L \geq \binom{m}{s}$.

۳. هر زیر ماتریس از $\mathbf{Y}$ شامل $s+1$ سیگنال، که همگی از یک دسته‌ی s تائی مشخص از ستون‌های $\mathbf{D}$ در نمایش خود استفاده می‌کنند، دارای رُتبه‌ای دقیقاً برابر با s است. هم‌چنین، برای زیرماتریسی که ستون‌های آن از اتم‌های مختلفی در نمایش خود استفاده می‌کنند، رُتبه برابر با $s+1$ است.

توجه کنید که اگرچه در یک مسأله‌ی آموزش دیکشنری، تعداد سیگنال‌های آموزشی خیلی بیش‌تر از تعداد اتم‌ها است؛ امّا با یک حساب سرانگشتی می‌توان فهمید که شرطی که این قضیه روی تعداد ستون‌های ماتریس $\mathbf{Y}$ می‌گذارد (شرط دوم) بسیار محدودکننده بوده و در عمل امکان‌پذیر نیست. اخیراً در [۳۴] مسأله‌ی بازیابی دیکشنری به عنوان مینیمم محلّی یک مسأله‌ی بهینه‌سازی، با در نظرگرفتن نُرم ℓ_1 به عنوان معیار تُنُک بودن (بر خلاف قضیه‌ی فوق که از شبه نُرم ℓ_0 استفاده می‌کند)، بررسی شده است. در این مقاله نشان داده شده است که تعداد داده‌های مورد نیاز برای این منظور رابطه‌ای به صورت $L \approx Cm \log m$ با تعداد اتم‌ها، یعنی m، دارد. در این عبارت C یک

[1] Matrix Factorization

عدد ثابت است.

برای اثبات قضیه‌ی ۳–۱، [۲] از یک روش سازنده[۱] استفاده کرده است؛ هرچند، به عنوان یک روش عملی حتّی در ابعاد کم نیز پیچیدگی بالائی دارد. به دلیل جزئیات زیاد، از بیان این اثبات خودداری می‌کنیم. در ادامه، بعنوان جایگزینی برای اثبات این قضیه، روشی ساده‌تر و البته مفهومی‌تر ذکر می‌کنیم[۲]. ابتدا توجه کنید که طبق فرض دومِ قضیه‌ی ۳–۱، به‌ازای هر زیرفضا، که ابعاد همگی برابر با s است، حداقل $s+1$ سیگنال وجود دارد و از طرفی، طبق فرض سومِ این قضیه، دقیقاً s سیگنالِ مستقل خطی در این بین وجود دارد. در نتیجه، هر زیرفضا با استفاده از s سیگنال مستقل خطیِ متناظرِ خود قابل توصیف است. برای بدست آوردن این زیرفضاها به این ترتیب عمل می‌کنیم که ابتدا یک دیکشنری اولّیه که شامل تمام ستون‌های ماتریس $\mathbf{Y}$ به استثنای $\mathbf{y}_1$ است در نظر می‌گیریم. می‌دانیم $\mathbf{y}_1$ به همراه s سیگنال مستقل خطیِ دیگر تشکیل یک زیرفضا می‌دهند. در نتیجه، این سیگنال را می‌توان بر حسب یک ترکیب خطی از این s سیگنال نوشت. برای این منظور، کافی است نمایش تُنُک $\mathbf{y}_1$ را در دیکشنری اولّیه‌ی فرض شده بدست آوریم. به این ترتیب، انتظار داریم که در صورت موفقیت الگوریتم بازیابی نمایش تُنُک مورد استفاده، نمایش بدست آمده برای این سیگنال دقیقاً شامل همان s سیگنال هم‌گروه خود باشد. بنابراین تا این‌جا یک زیرفضا را تشخیص دادیم. برای بدست آوردن بقیه‌ی زیرفضاها، دیکشنری را با حذف s سیگنال بدست آمده به‌روز کرده، روال قبل را برای $\mathbf{y}_2$ و به همین ترتیب برای بقیه‌ی سیگنال‌ها تکرار می‌کنیم. در نهایت، کافی است اتم‌های دیکشنری مورد سؤال را با اصلاح کردن این زیرفضاها، به طریقی مانند روش ذکر شده در [۳۳]، بدست آوریم.

۳–۵ الگوریتم‌های آموزش دیکشنری

در این بخش، تعدادی از معروف‌ترین الگوریتم‌های آموزش دیکشنری را که تاکنون معرفی شده‌اند، به اختصار مرور می‌کنیم. اساس این الگوریتم‌ها همانطور که در بخش ۳–۳ توضیح دادیم بر مبنای تکرار دو گام «نمایش تُنُک» و «به‌روز کردن دیکشنری» است. تفاوت بین این الگوریتم‌ها در روشی است که هر یک برای محاسبه‌ی نمایش تُنُک سیگنال‌ها و از آن مهم‌تر، برای به‌روز کردن دیکشنری استفاده می‌کند. بعبارت دیگر، تفاوت این الگوریتم‌ها

[۱] Constructive

[۲] این ایده به عنوان یک روش مستقل برای آموزش دیکشنری در [۳۳] مطرح شده است.

بیش‌تر در گام به‌روز کردن دیکشنری است.

۳-۵-۱ روش درستنمائی بیشینه

کارهای صورت گرفته در [۴۸، ۴۹] را می‌توان سرآغاز بحث آموزش دیکشنری‌های فوق کامل دانست. در این دو مقاله از «تخمین آماری» برای بدست آوردن دیکشنری استفاده شده است. در این روش، برای هر سیگنال آموزشی نوعیِ $\mathbf{y}$ مدل زیر فرض می‌شود:

$$\mathbf{y} = \mathbf{D}\mathbf{x} + \mathbf{n}, \tag{۳-۵}$$

که در آن $\mathbf{n}$ یک نویز سفید گوسی با میانگین صفر و واریانس σ^2 است. رهیافت این روش، استفاده از تخمین درستنمائی بیشینه[1] (ML) برای بدست آوردن دیکشنری است. تابع درست‌نمائی[2]، با فرض استقلال آماری ستون‌های $\mathbf{Y}$، برابر است با $P(\mathbf{Y}|\mathbf{D}) = \prod_{i=1}^{L} P(\mathbf{y}_i|\mathbf{D})$. تخمین ML دیکشنری بصورت زیر است:

$$\mathbf{D}^* = \arg\max_{\mathbf{D}} P(\mathbf{Y}|\mathbf{D}) \tag{۳-۶}$$

برای محاسبه‌ی $P(\mathbf{y}_i|\mathbf{D})$، ابتدا دقت کنید که در این‌جا $\mathbf{x}$ یک متغّیر نهفته[3] است؛ چرا که دانستن آن منوط به معلوم بودن $\mathbf{D}$ است. در نتیجه داریم:

$$P(\mathbf{y}_i|\mathbf{D}) = \int P(\mathbf{y}_i, \mathbf{x}|\mathbf{D})\,\mathrm{d}\mathbf{x} = \int P(\mathbf{y}_i|\mathbf{D}, \mathbf{x}).P(\mathbf{x})\,\mathrm{d}\mathbf{x}. \tag{۳-۷}$$

از طرفی، $P(\mathbf{y}_i|\mathbf{D}, \mathbf{x}) = C\exp\left\{\frac{1}{2\sigma^2}\|\mathbf{y}_i - \mathbf{D}\mathbf{x}\|_2^2\right\}$، که C یک عدد ثابت است. با فرض توزیع لاپلاسین برای $\mathbf{x}$ (جهت رسیدن به نمایش تُنُک) داریم:

$$P(\mathbf{y}_i|\mathbf{D}) = C\int \exp\left\{\frac{1}{2\sigma^2}\|\mathbf{y}_i - \mathbf{D}\mathbf{x}\|_2^2\right\}.\exp\left\{\lambda\|\mathbf{x}\|_1\right\}\mathrm{d}\mathbf{x}. \tag{۳-۸}$$

به دلیل اینکه محاسبه‌ی انتگرال فوق دشوار است، [۴۸] از ماکزیمم $P(\mathbf{y}_i|\mathbf{D}, \mathbf{x})$ استفاده کرده است. در اینصورت برای مسأله‌ی نهائی داریم:

$$\mathbf{D}^* = \arg\max_{\mathbf{D}} \sum_{i=1}^{L} \max_{\mathbf{x}_i} P(\mathbf{y}_i, \mathbf{x}_i|\mathbf{D}) = \arg\min_{\mathbf{D}} \sum_{i=1}^{L} \min_{\mathbf{x}_i} \left\{\|\mathbf{y}_i - \mathbf{D}\mathbf{x}_i\|_2^2 + \lambda\|\mathbf{x}_i\|_1\right\} \tag{۳-۹}$$

رهیافت حلّ مسأله‌ی فوق همان استراتژی کلّی دو گامی است که پیش‌تر توضیح داده شد. در [۴۸، ۴۹] برای بدست آوردن نمایش تُنُک سیگنال‌ها در گام اول، از الگوریتم گرادیان کاهشی استفاده شده است. در گام دوم نیز با استفاده از الگوریتم گرادیان کاهشی، دیکشنری بصورت زیر به‌روز می‌شود:

[1] Maximum Liklihood

[2] Liklihood Function

[3] Hidden Variable

$$\mathbf{D}^{(t+1)} = \mathbf{D}^{(t)} - \eta(\mathbf{D}^{(t)}\mathbf{X} - \mathbf{Y})\mathbf{X}^T. \quad (۱۰-۳)$$

توجه کنید که برای هر ماتریسِ (قطری) مثبت معینِ $\mathbf{\Lambda}$ داریم $\mathbf{D\Lambda\Lambda}^{-1}\mathbf{X} = \mathbf{DX}$. لذا در گام بدست آوردن نمایش تُنُک، چون ضرایبِ بسیار کوچک مطلوب‌اند، در نتیجه، الگوریتم در جهتی پیش می‌رود که اندازه‌ی درایه‌های $\mathbf{X}$ تا حدّ امکان کوچک شود و در عوض، نُرم ستون‌های $\mathbf{D}$ افزایش یابد. برای جلوگیری از بروز این اتفاق، بعد از به‌روز کردن دیکشنری، اتم‌ها نُرمالیزه می‌شوند.

لازم به ذکر است که به عنوان یک روش تخمین آماری دیگر، رهیافت «حداکثر احتمال پسین»[۱] (MAP) نیز در [۴۰] برای آموزش دیکشنری استفاده شده است. در تخمین MAP به جای تابع درست‌نمائی، تابع احتمال پسین ماکزیمم می‌شود. این تابع با توجه به قاعده‌ی بیز[۲] بصورت $P(\mathbf{D}|\mathbf{Y}) \propto P(\mathbf{Y}|\mathbf{D})P(\mathbf{D})$ است. بنابراین، لازم است یک توزیع پیشینِ $P(\mathbf{D})$ برای دیکشنری فرض کنیم. در [۴۰] توزیع‌های مختلفی برای دیکشنری فرض شده است، از جمله اینکه نُرم فروبینیوس دیکشنری یا نُرم اتم‌ها واحد است. برای بدست آوردن نمایش تُنُک نیز از الگوریتم FOCUSS استفاده شده است. برای جزئیات بیشتر به مرجع مربوطه مراجعه کنید.

۳-۵-۲ الگوریتم MOD

الگوریتم جهت‌های بهینه یا به اختصار MOD[۳] یکی از ابتدائی‌ترین و البته ساده‌ترین الگوریتم‌هائی است که در سال ۱۹۹۹ معرفی شد [۲۹]. در گام بدست آوردن نمایش تُنُک، از هر الگوریتمی می‌توان استفاده کرد (در [۲۹] از الگوریتم OMP استفاده شده است). در گام دوم و با استفاده از ماتریس ضرایب که در گام نخست بدست آمده است، اتم‌ها در جهتی تغییر داده می‌شوند، که خطای کلّی نمایش یعنی، $E = \|\mathbf{Y} - \mathbf{DX}\|_F^2$ ، حداقل شود. این کار معادل با حلّ مسأله‌ی (۳-۴) با فرض $\mathbf{X}$ ثابت است. برای این منظور، کافی است مشتق این عبارت را نسبت به $\mathbf{D}$ برابر با صفر قرار دهیم. در اینصورت بدست می‌آوریم $(\mathbf{Y} - \mathbf{DX})\mathbf{X}^T = 0$. نهایتاً برای دیکشنری به‌روز شده داریم:

$$\mathbf{D} = \mathbf{YX}^T(\mathbf{XX}^T)^{-1} = \mathbf{YX}^\dagger. \quad (۱۱-۳)$$

[۱] MaximumA-Posteriori Probability

[۲] Bayes's rule

[۳] Method of Optimal Directions

دقت کنید که بعد از به‌روز کردن دیکشنری لازم است ستون‌های آن نُرمالیزه شود. ممکن است طی پیشروی الگوریتم در برخی تکرارها ماتریس $\mathbf{X}\mathbf{X}^T$ بد حالت[۱] شود. این وضعیت می‌تواند اثر سوئی روی دیکشنریِ به‌روز شده داشته باشد. برای رفع این مشکل، ماتریس $\lambda\mathbf{I}$ به $\mathbf{X}\mathbf{X}^T$ اضافه می‌شود، که در آن λ یک عدد مثبت کوچک است. در اینصورت داریم:

$$\mathbf{D} = \mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}. \qquad (۱۲-۳)$$

به راحتی می‌توان نشان داد که دیکشنری فوق در واقع جواب مسأله‌ی زیر است:

$$\min_{\mathbf{D}} \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 + \lambda\|\mathbf{D}\|_F^2. \qquad (۱۳-۳)$$

تفاوت MOD با روش ML در نحوه‌ی انجام دو گامِ نمایش تُنُک و به‌روز کردن دیکشنری است. MOD از الگوریتم OMP در گام نمایش تُنُک استفاده می‌کند که به مراتب از الگوریتم گرادیان کاهشی مورد استفاده در ML کارآتر است. بعلاوه، در گام به‌روز کردن دیکشنری، MOD «بهترین دیکشنری» را با مینیمم کردن خطای کلّی بدست می‌آورد؛ حال آنکه روش ML از الگوریتم تکراری گرادیان کاهشی برای این منظور استفاده می‌کند که دیکشنری بدست آمده به این طریق لزوماً بهینه نیست.

۳-۵-۳ الگوریتم K-SVD

الگوریتم K-SVD [۱] یکی از موفق‌ترین الگوریتم‌های آموزشی دیکشنری بوده که نسبت به بقیه‌ی الگوریتم‌ها رابطه‌ای نزدیک‌تر با الگوریتم K-means دارد. K-SVD در گام نمایش تُنُک از الگوریتم OMP استفاده می‌کند. انتخاب این الگوریتم به دلیل سرعت بالای آن نسبت به بقیه‌ی الگوریتم‌های کُدینگ تُنُک است؛ اگرچه به دلیل ماهیت حریص آن ممکن است جواب‌های نادرستی داشته باشد.

برای به‌روز کردن دیکشنری، بر خلاف MOD که کلّ اتم‌ها را یک‌جا به روز می‌کند، K-SVD اتم‌ها را یک به یک و به طور متوالی به‌روز می‌کند. به یاد بیاورید که در K-means هم، بُردارهای کُد یک به یک به روز می‌شوند. برای به‌روز کردن اتم $\mathbf{d}_k$، بقیه‌ی اتم‌ها را ثابت می‌گیریم. در اینصورت داریم:

$$E = \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 = \|\mathbf{Y} - \sum_{i=1}^{m}\mathbf{d}_i\mathbf{x}_T^i\|_F^2 = \|(\mathbf{Y} - \sum_{i\neq k}^{m}\mathbf{d}_i\mathbf{x}_T^i) - \mathbf{d}_k\mathbf{x}_T^k\|_F^2 = \|\mathbf{E}_k - \mathbf{d}_k\mathbf{x}_T^k\|_F^2, \qquad (۱۴-۳)$$

که در آن منظور از $\mathbf{x}_T^i$ سطر i اُم ماتریس $\mathbf{X}$ است. در نتیجه، برای به‌روز کردن اتم $\mathbf{d}_k$ باید مسأله‌ی زیر را حل کنیم:

[۱] Ill-conditioned

$$\min_{\mathbf{d}} \|\mathbf{E}_k - \mathbf{d}\mathbf{x}_T^k\|_F^2 \quad \text{subject to} \quad \|\mathbf{d}\|_2 = 1. \tag{۱۵–۳}$$

با اندکی محاسبات، جواب این مسأله به صورت زیر بدست می‌آید:

$$\mathbf{d}_k \leftarrow \text{normalize}(\mathbf{E}_k(\mathbf{x}_T^k)^T) \tag{۱۶–۳}$$

که در آن $\text{normalize}(\mathbf{d}) \triangleq \mathbf{d}/\|\mathbf{d}\|_2$. ایده‌ی K-SVD امّا فراتر از این است. K-SVD علاوه بر هر اتم، سطر متناظر آن در ماتریس ضرایب را نیز به‌روز می‌کند؛ و بعلاوه، در به‌روز کردن هر اتم تنها سیگنال‌هائی به کار می‌روند که قبلاً (در گام اوّل) از این اتم در نمایش خود استفاده کرده‌اند. برای توضیح بیش‌تر، اولاً دقت کنید که اگر در محاسبه‌ی ماتریس $\mathbf{E}_k$، از مقادیر به‌روز شده‌ی $\mathbf{d}_i$ و $\mathbf{x}_T^i$ ها استفاده کنیم، سرعت همگرائی الگوریتم بیش‌تر خواهد شد. به عنوان مثال، بعد از به‌روز کردن $\mathbf{d}_1$ و $\mathbf{x}_T^1$، برای به‌روز کردن اتم دوم و سطر متناظر آن، از مقادیر به‌روز شده‌ی $\mathbf{d}_1$ و $\mathbf{x}_T^1$ در محاسبه‌ی $\mathbf{E}_2$ استفاده کنیم و به همین ترتیب برای بقیه اتم‌ها و سطر متناظرشان. مسأله‌ی به روز کردن همزمان اتم $\mathbf{d}_k$ و $\mathbf{x}_T^k$ به صورت زیر است:

$$\min_{\mathbf{d},\mathbf{x}_T} \|\mathbf{E}_k - \mathbf{d}\mathbf{x}_T\|_F^2. \tag{۱۷–۳}$$

مسأله‌ی فوق در واقع تقریب رُتبه–۱ ماتریس $\mathbf{E}_k$ است. به طور کلّی، اگر تجزیه به مقادیر تکینِ[1] (SVD) ماتریس دلخواه $\mathbf{A} \in \mathbb{R}^{n\times m}$ بصورت $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ باشد، تقریب رتبه–r این ماتریس که $r \le \min(n,m)$، بصورت $\mathbf{A}_r = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$ است.

مشکلی که در اینجا بوجود می‌آید این است که طی این فرآیند به احتمال زیاد تمام درایه‌های بُردار $\mathbf{x}_T^k$ پُر خواهد شد. به همین دلیل تنها از سیگنال‌هائی استفاده می‌شود که در گام قبلی از اتم $\mathbf{d}_k$ در نمایش خود استفاده کرده‌اند (به شباهت این کار با به‌روز کردن بُردارهای کُد در K-means دقت کنید). به این ترتیب، تنها درایه‌های غیر صفر بُردار $\mathbf{x}_T^k$ به‌روز شده و بقیه‌ی درایه‌های آن صفر می‌مانند. اگر تعریف کنیم $\omega_k = \{i : 1 \le i \le m,\ \mathbf{x}_T^k(i) \neq 0\}$، در نهایت مسأله‌ی زیر را برای به روز کردن اتم $\mathbf{d}_k$ و درایه‌های غیر صفر سطر $\mathbf{x}_T^k$ داریم:

$$\min_{\mathbf{d},\mathbf{x}_r} \|\mathbf{E}_k^{\omega_k} - \mathbf{d}\mathbf{x}_r\|_F^2 \quad \text{subject to} \quad \|\mathbf{d}\|_2 = 1, \tag{۱۸–۳}$$

که در آن $\mathbf{E}_k^{\omega_k}$ شامل تنها ستون‌های متناظر با ω_k از ماتریس $\mathbf{E}_k$ و $\mathbf{x}_r$ نیز بُرداری به طول $|\omega_k|$ است. اگر تجزیه به مقادیر تکینِ $\mathbf{E}_k^{\omega_k}$ بصورت $\mathbf{E}_k^{\omega_k} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ باشد، داریم:

[1] Singular Value Decomposition

$$\mathbf{d}_k \leftarrow \mathbf{u}_1, \quad \mathbf{x}_T^k(\omega_k) \leftarrow \sigma_1 \mathbf{v}_1, \tag{۱۹-۳}$$

که σ_1 بزرگترین مقدار تکین است. توجه کنید که چون نُرم ستون‌های $\mathbf{U}$ واحد است، بنابراین نیازی به نُرمالیزه کردن $\mathbf{d}_k$ نیست. همه‌ی اتم‌های دیکشنری به این ترتیب به‌روز می‌شوند. وجه تسمیه‌ی K-SVD نیز مشابه K-means به خاطر این است که به تعداد اتم‌ها (K، ولی در نمادگذاری ما m) SVD انجام می‌شود.

۳-۵-۴ الگوریتم MM-DL[1]

در زیربخشِ ۲-۵-۲-۱ دیدیم که ایده‌ی الگوریتم‌های IST استفاده از روش تابع جایگزین یا MM است. این ایده در [۶۲] برای آموزش دیکشنری بکار رفته است. در این روش، مسأله‌ی زیر برای آموزش دیکشنری استفاده شده است:

$$\min_{\mathbf{D}\in\mathcal{D},\mathbf{X}} \|\mathbf{Y}-\mathbf{D}\mathbf{X}\|_F^2 + \lambda\|\mathbf{X}\|_1, \tag{۲۰-۳}$$

که در آن $\|\mathbf{X}\|_1 \triangleq \sum_i \|\mathbf{x}_i\|_1$ و $\mathcal{D}$ مجموعه‌ی مُجاز[2] دیکشنری است که به دو صورت زیر درنظر گرفته شده است:

$$\mathcal{D}_F = \left\{\mathbf{D}\in\mathbb{R}^{n\times m} : \ \|\mathbf{D}\|_F^2 \le c_F\right\} \tag{۲۱-۳}$$

که نُرم فروبینیوس دیکشنری را محدود می‌کند و

$$\mathcal{D}_C = \left\{\mathbf{D}\in\mathbb{R}^{n\times m} : \ \|\mathbf{d}_i\|_2^2 \le c_C\right\} \tag{۲۲-۳}$$

که نُرم اتم‌ها را محدود می‌کند. دقت کنید که این دو مجموعه محدّب بوده و لذا مسأله‌ی به‌روز کردن دیکشنری (که در آن $\mathbf{X}$ ثابت است) محدّب خواهد بود. برای بدست آوردن $\mathbf{X}$ در گام اول، نویسنده‌های [۶۲] با طی روالی مشابه زیربخشِ ۲-۵-۲-۱، فرمول تکراری زیر را بدست آورده اند[3]:

$$\mathbf{Z} = \frac{1}{c_X}(\mathbf{D}^T\mathbf{Y} + (c_X\mathbf{I} - \mathbf{D}^T\mathbf{D})\mathbf{X}^{(t)}), \quad \mathbf{X}^{(t+1)} = \mathcal{S}_\lambda(\mathbf{Z}), \tag{۲۳-۳}$$

که در آن $c_X > \lambda_{\max}(\mathbf{D}^T\mathbf{D})$ یک عدد ثابت است. در گام دوم، دیکشنری بصورت تکراریِ زیر به‌روز می‌شود:

---

[1] Majorization Minimization Dictionary Learning

[2] Admissible set

[3] اگرچه نویسنده‌های این مقاله این فرمول را «فُرم ماتریسی نمایش تُنُک» نامیده‌اند، امّا این معادله براحتی بر حسب ستون‌های $\mathbf{X}$ جداپذیر است؛ یعنی نمایش تُنُک هر سیگنال را می‌توان از همان «فُرم بُرداری» زیربخشِ ۲-۵-۲-۱ بدست آورده و نهایتاً ستون‌های $\mathbf{X}$ را تشکیل داد. ولی نکته‌ی مهمی که در فُرم ماتریسی وجود دارد این است که لازم نیست برای بدست آوردن نمایش تُنُک سیگنال‌ها دو ماتریس $\mathbf{D}^T\mathbf{Y}$ و $\mathbf{D}^T\mathbf{D}$ را هر بار حساب کرد؛ بلکه تنها کافی است «یک‌بار» محاسبه و ذخیره شوند. ظاهراً نویسنده‌ها به این نکته توجه نداشته‌اند؛ چرا که حتّی در پیاده‌سازی معادله‌ی (۳-۲۳) نیز دقت نکرده‌اند که محاسبه‌ی یک‌بارِ این دو ماتریس برای انجام این تکرارها کافی است.

$$\mathbf{D}^{(t+1)} = \arg\min_{\mathbf{D}\in\mathcal{D}} \left\{ \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 + d(\mathbf{D}, \mathbf{D}^{(t)}) \right\}, \tag{۲۴–۳}$$

که $d(\mathbf{D}, \mathbf{D}^{(t)}) = c_D\|\mathbf{D} - \mathbf{D}^{(t)}\|_F^2 - \|\mathbf{D}\mathbf{X} - \mathbf{D}^{(t)}\mathbf{X}\|_F^2$ و $c_D > \lambda_{\max}(\mathbf{X}^T\mathbf{X})$ یک عدد ثابت است. با حلّ مسأله‌ی (۳–۲۴) با استفاده از ضرایب لاگرانژ (برای حذف قیدِ $\mathcal{D}$) بدست می‌آوریم:

$$\mathbf{C} = \frac{1}{c_D}(\mathbf{Y}\mathbf{X}^T + \mathbf{D}^{(t)}(c_D\mathbf{I} - \mathbf{X}\mathbf{X}^T)), \quad \mathbf{D}^{(t+1)} = \mathcal{P}(\mathbf{C}), \tag{۲۵–۳}$$

که $\mathcal{P}$ عملگرِ تصویر کردن روی مجموعه‌ی $\mathcal{D}$ است. با تغییر آرایش جملات در معادله‌ی (۳–۲۵) براحتی می‌توان فهمید که الگوریتم به‌روز کردن دیکشنری چیزی نیست جز یک الگوریتم گرادیان–تصویر با طول گام ثابت و برابر با $\mu = 1/c_D$.

۳–۵–۵ الگوریتم RLS-DL[1]

الگوریتم‌هائی که تا این‌جا بررسی کردیم، در گام به‌روز کردن دیکشنری، «تمام داده‌های آموزشی» را یک‌جا پردازش می‌کنند. به همین دلیل، به این دسته از الگوریتم‌ها اصطلاحاً «الگوریتم‌های مبتنی بر گروه»[2] می‌گویند. دسته‌ی دیگری از الگوریتم‌ها اخیراً پیشنهاد شده است که مبتنی بر «پردازش یکی یکیِ داده‌های آموزشی» هستند [۴۱، ۴۴، ۵۳]. ایده‌ی کلّی این دسته از الگوریتم‌ها، بکارگیری «آموزش آنلاین»[3]، که در حوزه‌ی آموزش ماشین[4] خیلی معروف است، برای آموزش دیکشنری است. در نتیجه، با پردازش متوالی داده‌ها، دیکشنری به طور پیوسته به‌روز می‌شود. مزیت این دسته از الگوریتم‌ها نسبت به الگوریتم‌های دیگر این است که چون داده‌ها را یکی یکی پردازش می‌کنند، بنابراین در کاربردهائی که حجم داده‌ها بسیار وسیع است، براحتی مورد استفاده قرار می‌گیرند. در این زیربخش، ما الگوریتم RLS-DL [۵۳] را از میان این دسته از الگوریتم‌ها به اختصار معرفی می‌کنیم.

دو ماتریس $\mathbf{Y}_t = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_t]$ و $\mathbf{X}_t = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t]$ را بعنوان ماتریس داده‌هائی که تاکنون پردازش شده‌اند، و ماتریسِ شامل نمایش تُنُک آن‌ها تعریف می‌کنیم. دیکشنری بهینه برای این دسته از داده‌ها طبق رابطه‌ی (۳–۱۱) برابر است با:

$$\mathbf{D}_t = (\mathbf{Y}_t\mathbf{X}_t^T)(\mathbf{X}_t\mathbf{X}_t^T)^{-1} = \mathbf{B}_t\mathbf{C}_t, \tag{۲۶–۳}$$

[1] Recursive Least Squares Dictionary Learning

[2] Batch-based algorithms

[3] Online Learning

[4] Machine Learning

که $\mathbf{B}_t = \mathbf{Y}_t\mathbf{X}_t^T = \sum_{i=1}^{t} \mathbf{y}_i\mathbf{x}_i^T$ و $\mathbf{C}_t^{-1} = \mathbf{X}_t\mathbf{X}_t^T = \sum_{i=1}^{t} \mathbf{x}_i\mathbf{x}_i^T$. حال فرض کنید می‌خواهیم داده‌ی $\mathbf{y} = \mathbf{y}_{t+1}$ را پردازش کنیم. ابتدا نمایش تُنُک $\mathbf{y}$، یعنی $\mathbf{x}$، را با استفاده از $\mathbf{D}_t$ بدست می‌آوریم. در اینصورت برای دیکشنریِ به‌روز شده داریم: $\mathbf{D}_{t+1} = \mathbf{B}_{t+1}\mathbf{C}_{t+1}$ که $\mathbf{B}_{t+1} = \mathbf{B}_t + \mathbf{y}\mathbf{x}^T$ و $\mathbf{C}_{t+1}^{-1} = \mathbf{C}_t^{-1} + \mathbf{x}\mathbf{x}^T$. ایده‌ی اصلی این الگوریتم، که ابتدا در الگوریتم RLS در بحث فیلترهای وفقی[۱] استفاده شده است، این است که چون ماتریسِ $\mathbf{C}_{t+1}^{-1}$ بصورت رُتبه-۱ به‌روز می‌شود، برای جلوگیری از محاسبه‌ی مستقیم این معکوس، می‌توان از لِم معکوس ماتریس[۲] برای این منظور استفاده کرد که نهایتاً به عبارت زیر می‌انجامد:

$$\mathbf{C}_{t+1} = \mathbf{C}_t - \frac{\mathbf{C}_t\mathbf{x}\mathbf{x}^T\mathbf{C}_t}{\mathbf{x}^T\mathbf{C}_t\mathbf{x} + 1}. \qquad (۲۷-۳)$$

در نهایت، با انجام کمی محاسبات جبری، رابطه‌ی بهبود دیکشنری بصورت زیر بدست می‌آید:

$$\mathbf{D}_{t+1} = \mathbf{D}_t + \alpha\mathbf{r}\mathbf{u}^T, \qquad (۲۸-۳)$$

که $\mathbf{r} = \mathbf{y} - \mathbf{D}_t\mathbf{x}$، $\mathbf{u} = \mathbf{C}_t\mathbf{x}$ و $\alpha = (\mathbf{x}^T\mathbf{C}_t\mathbf{x} + 1)^{-1}$. برای کاهش اثر نامطلوب دیکشنری‌های قبلی، مشابه الگوریتم RLS، یک ضریب فراموشیِ[۳] λ نیز معرفی شده است. در اینصورت بدون ذکر جزئیات، روابط نهائی مشابه روابط فوق است، با این تفاوت که ماتریس $\mathbf{C}_t$ با $\lambda^{-1}\mathbf{C}_t$ جایگزین می‌شود. دقت کنید که دو گام «نمایش تُنُک» و «به‌روز کردن دیکشنری» در این دسته از الگوریتم‌ها بصورت تودرتو و کلاف‌مانند انجام می‌شود. به بیان دیگر، بعد از این که همه‌ی داده‌های آموزشی را پردازش کردیم، معادل آن است که «یک‌بار» این دو گام را انجام داده‌ایم[۴].

لازم به توضیح است که یک تفاوت اساسیِ کاربردِ آموزش آنلاین در آموزش دیکشنری، با کاربردهای دیگرِ آن مانند الگوریتم RLS در حوزه‌ی فیلترهای وفقی این است که بعنوان مثال در دسته‌ی دوم، تنها مجهول ما بُردار ضرایب فیلتر است که می‌خواهیم با معلوم بودن تعداد زیادی ورودی و خروجی از فیلتر، این ضرایب را طی یک فرآیند آموزشی بدست آوریم. در بحث آموزش دیکشنری امّا اگر $\mathbf{x}$ را ورودی یک فیلتر، $\mathbf{y} = \mathbf{D}\mathbf{x}$ را خروجی فیلتر و $\mathbf{D}$ را نیز به عنوان ضرایب فیلتر فرض کنیم، در اینصورت تنها معلوم ما بُردار $\mathbf{y}$ خواهد بود؛ زیرا بُردار $\mathbf{x}$ وابسته‌ی به دیکشنری بوده و پابه‌پای آن بدست می‌آید.

[۱] Adaptive Filters

[۲] Matrix Inversion Lemma

[۳] Forgetting Factor

[۴] اگر چه در [۵۳] بطور صریح این موضوع ذکر نشده است.

۳-۶ جمع‌بندی

در این فصل، مسأله‌ی آموزش دیکشنری را بررسی کردیم. دیدیم که الگوریتم‌های آموزش دیکشنری در واقع تعمیمی از الگوریتم معروف K-means هستند، که با انجام دو گام «بدست آوردن نمایش تُنُک» و «به‌روز کردن دیکشنری» سعی دارند برای یک‌سری داده‌ی آموزشی، یک دیکشنری تُنُک‌کننده (که اتم‌های آن به بهترین وجه، ویژگی‌های برجسته‌ی این داده‌ها را نشان می‌دهند) آموزش دهند. در ادامه، به مسأله‌ی آموزش دیکشنری به عنوان تجزیه‌ی ماتریس داده‌ها بصورت ضرب یک ماتریس با ستون‌های نُرمالیزه و یک ماتریس با ستون‌های تُنُک نگاه کردیم و یکتائی چنین تجزیه‌ای را با استفاده از مراجع موجود بیان کردیم. در خاتمه نیز چند الگوریتم معروف آموزش دیکشنری را بررسی کردیم. دیدیم که اصول همگی مبتنی بر انجام همان دو گام کلّی است و آنچه باعث تفاوت این الگوریتم‌ها می‌شود، نحوه‌ی انجام این دو گام است. در فصل بعد، نویززدائی از تصاویر با استفاده از نمایش تُنُک را بررسی می‌کنیم.

فصل ۴

کاربرد پردازش تُنُک در نویززدائی تصاویر

۴-۱ مقدمه

در دو فصل گذشته، نمایش تُنُک سیگنال‌ها را بررسی کردیم. دیدیم که دو مسأله‌ی اساسی در این بحث وجود دارد. یکی ارائه‌ی الگوریتم‌های کارآ برای بدست آوردن نمایش تُنک سیگنال‌ها است و دیگری طراحی یا آموزش یک دیکشنری تُنُک کننده برای پردازش یک کلاس مشخص از داده‌ها، مانند تصاویر طبیعی. مسأله‌ی اول را در فصل دوم بررسی کردیم. در این فصل، بعد از معرفی مباحث پایه‌ای نمایش تُنُک، تعدادی از الگوریتم‌های عملی برای بازیابی این نمایش را معرفی کردیم. در فصل سوم نیز موضوع مهم «آموزش دیکشنری» را به همراه تعدادی از الگوریتم‌های موجود برای این کار معرفی کردیم. بنابراین، تا این‌جا تقریباً مباحث اساسی موجود در پردازش تُنُک سیگنال‌ها را بررسی کرده‌ایم. در فصل حاضر، به عنوان یکی از کاربردهای پردازش تُنُک سیگنال‌ها در حوزه‌ی تصاویر، بحث نویززدائی از تصاویر را بررسی می‌کنیم. توجه کنید که مسأله‌ی نویززدائی از تصاویر بیش‌تر از نیم قرن است که موضوع تحقیقات بهبود تصاویر در سراسر جهان است؛ لذا تاکنون الگوریتم‌های بسیار زیادی در این زمینه پیشنهاد شده است [۳۲، ۲۵]. هدف از این فصل بررسی صرفِ نویززدائی از تصاویر نیست؛ چرا که این خود می‌تواند موضوع یک پایان‌نامه باشد. در این‌جا صرفاً به عنوان یک رهیافت جدید [۲۵] که در سال ۲۰۰۶ معرفی شده است، نویززدائی از تصاویر را به کمک پردازش تُنُک سیگنال‌ها بررسی خواهیم کرد.

در ادامه‌ی این فصل، منظور ما از سیگنال، همان تصویر خواهد بود و بعلاوه، یک تصویر دو بُعدی را با یک

بُردار که از زیر هم چیده شدن درایه‌های آن (با یک ترتیب قراردادی) بدست می‌آید، نشان خواهیم داد. افزون بر این، چون عموماً طول بُردارهای بدست آمده بسیار بالا بوده و پردازش تُنُکِ تصاویر (لااقل با الگوریتم‌های فعلی) قادر به کارکردن با چنین بُردارهای طویلی نیست، تصویر را به چندین بلوک تقسیم کرده، عملیات نویززدائی را روی هر بلوک انجام داده و در نهایت با متوسط‌گیری این بلوک‌ها [۲۵]، تصویر نهائی را بدست می‌آوریم.

۴-۲ معرفی مسأله‌ی نویززدائی

مسأله‌ی نویززدائی ساده‌ترین حالت یک مسأله‌ی معکوس است [۲۶، ۲۷]. حالت کلّی این دسته از مسائل به این ترتیب است که سیگنال تمیزِ $\mathbf{y}_0$ تحت تبدیلِ $\mathbf{H}$ و در حضور نویزِ سفید گوسی جمع شونده[۱] (AWGN) اندازه‌گیری می‌شود. در نتیجه، سیگنال اندازه‌گیری شده بصورت زیر است:

$$\mathbf{y} = \mathbf{H}\mathbf{y}_0 + \mathbf{n}. \qquad (۱-۴)$$

بسته به کاربرد، ماتریسِ تبدیلِ $\mathbf{H}$ تعابیر مختلفی دارد. بعنوان مثال، در مسأله‌ی برطرف کردن محوشدگی[۲]، این ماتریس مدل‌کننده‌ی عمل تارشدگی است؛ یا در بزرگ‌نمائی[۳]، این ماتریس نشان‌دهنده‌ی عمل کوچک کردن[۴] (و احتمالاً تارشدگی) است. در نویززدائی امّا داریم $\mathbf{H} = \mathbf{I}$. بنابراین، مدل ما در این وضعیت تبدیل می‌شود به:

$$\mathbf{y} = \mathbf{y}_0 + \mathbf{n}. \qquad (۲-۴)$$

هدف نویززدائی سپس تخمین سیگنال تمیزِ $\mathbf{y}_0$ با مشاهده‌ی نسخه‌ی نویزی آن، یعنی $\mathbf{y}$، است. از دید تئوری تخمین، اگر سیگنال $\mathbf{y}_0$ را ثابت بگیریم (یعنی توزیعی برای آن فرض نکنیم؛ مانند تخمین ML)، در اینصورت چون $\mathbf{n} \sim \mathcal{N}(\mathbf{o}, \sigma^2\mathbf{I})$، در نتیجه $\mathbf{y} \sim \mathcal{N}(\mathbf{y}_0, \sigma^2\mathbf{I})$. بنابراین از این نگاه، مسأله‌ی نویززدائی عبارتست از تخمین میانگینِ متغیّر تصادفی $\mathbf{y}$ با مشاهده‌ی یک تحقّق[۵] از آن.

---

[۱] Additive White Gaussian Noise

[۲] Deblurring

[۳] Superresolution

[۴] Down Sampling

[۵] Realization

۴-۳ مرور مختصر روش‌های موجود

همانطور که پیش‌تر ذکر شد، روش‌های زیادی برای نویززدائی از تصاویر وجود دارد. از جمله روش‌های کلاسیک می‌توان به فیلترهای حوزه‌ی مکان، شامل انواع فیلترهای میانگین‌گیر (میانگین هندسی، حسابی و ...)، فیلترهای آماری شامل فیلترهای حداقل، حداکثر و میانه‌گیر، فیلترهای حوزه‌ی فرکانس و فیلتر وینر اشاره کرد [۳۲]. دسته‌ی دیگری از روش‌ها مبتنی بر تبدیل سیگنال بوده که به «روش‌های حوزه‌ی تبدیل»[۱] معروف‌اند. ایده‌ی کلّی این روش‌ها عبارتست از اعمالِ یک تبدیلِ مناسب بر روی سیگنال نویزی، سپس آستانه‌گذاری ضرایب تبدیل و در نهایت اعمال تبدیل معکوس برای بدست آوردن تخمینی از سیگنال تمیز. به بیان دقیق‌تر، اگر تبدیل مورد استفاده را با عملگر $\mathbf{T}$ و عملیات آستانه‌گذاری را با عملگر $\mathcal{A}$ نشان دهیم، در اینصورت رهیافت روش‌های حوزه‌ی تبدیل بصورت زیر است:

$$\mathbf{x} = \mathbf{T}(\mathbf{y}) \longrightarrow \hat{\mathbf{x}} = \mathcal{A}(\mathbf{x}) \longrightarrow \hat{\mathbf{y}}_{\circ} = \mathbf{T}^{-1}(\hat{\mathbf{x}}) \qquad (۳-۴)$$

برای توجیه این عملیات ابتدا بیاد آورید که طبق آنچه در فصل ۱ گفتیم، بسیاری از مسائل پردازش سیگنال، خصوصاً مسائل معکوس، مبتنی بر فرض یک مدلِ خوب برای سیگنال تحت بررسی است. این مدل (که در تئوری تخمین معادل با فرض یک توزیع احتمالاتی پیشین[۲] است) در حقیقت متمایزکننده‌ی سیگنال مطلوب از سیگنال نامطلوب (در این‌جا نویز) است. هرچقدر این مدل برای سیگنالِ مطلوب غنی‌تر باشد، کیفیت نهائی جواب نیز بهتر خواهد بود. با این توضیحات، ایده‌ی روش‌های حوزه‌ی تبدیل این است که سیگنال مطلوب در تبدیلِ (دیکشنریِ) مورد استفاده نمایشی تُنُک دارد و برعکس؛ نویز در این دیکشنری بخوبی نمایش داده نمی‌شود. در نتیجه، انرژی نویز در تعداد زیادی ضریب (یا روی تعداد زیادی اتم) گسترده خواهد بود و برعکس؛ انرژی سیگنال در تعداد کمی ضریب متمرکز است. بنابراین، با آستانه‌گذاری روی ضرایب تبدیل انتظار داریم درصد زیادی از انرژی نویز تضعیف شود.

نخستین بار، ایده‌ی تُنُک بودن نمایش سیگنال در تبدیل ویولت یکانی[۳] مورد استفاده قرار گرفت که نتیجه‌ی آن پیدایش الگوریتم‌های انقباض[۴] بود [۲۲]. اگرچه بعد از آن، نگاه‌ها به سمت استفاده از دیکشنری‌های فوق کامل

[۱] Transform Domain Methods

[۲] Priori Probability Distribution

[۳] Unitary Wavelet Transform

[۴] Shrinkage

سوق پیدا کرد؛ امّا همانطور که در فصل قبل توضیح دادیم، این تبدیل‌ها اغلب نمی‌توانند نمایشِ خوب یا تُنُکی برای سیگنالِ مطلوب ارائه دهند. در نتیجه، ایده‌ی «آموزش دیکشنری» برای نویززدائی از تصاویر در سال ۲۰۰۶ و در [۲۵] مطرح شد. در بخش بعدی این رهیافت را بررسی می‌کنیم.

قبل از اینکه وارد بخش بعدی شویم، خوب است اشاره‌ای داشته باشیم به الگوریتم تطبیق بلوک و فیلترینگِ سه بُعدی[۱] (BM3D) [۱۵] که جزء بهترین روش‌های موجود برای نویززدائی است. الگوریتم BM3D پیکسل به پیکسل روی تصویر عمل می‌کند؛ به این ترتیب که برای هر پیکسل، بلوک پیرامون آن را استخراج می‌کند. سپس روی کلّ تصویر حرکت کرده و بلوک‌های شبیه به آن (به مفهوم نُرم ℓ_2) را پیدا کرده، بصورت یک ماتریس سه بُعدی روی هم قرار می‌دهد. برای کاهش اثر نامطلوب نویز در محاسبه‌ی فاصله، بلوک‌های تصویر ابتدا به کمک آستانه‌گذاری تا حدّی نویززدائی می‌شوند. ایده‌ی این الگوریتم سپس اعمال یک تبدیل سه بُعدیِ DCT روی ماتریس بدست آمده است. دقت کنید که به دلیل شباهت زیاد بلوک‌های موجود در این ماتریس، انتظار داریم نمایش بسیار تُنُکی حاصل شود. نهایتاً مشابه روال کلّی روش‌های حوزه‌ی تبدیل، ضرایب بدست آمده آستانه‌گذاری شده و تبدیل عکس گرفته می‌شود. این کار برای همه‌ی پیکسل‌های تصویر انجام شده و در آخر، همه‌ی بلوک‌ها (با اعمال متوسط‌گیری) کنار هم قرار داده می‌شوند.

یک روش دیگر توجیه عملکرد این الگوریتم به این ترتیب است که ابتدا توجه کنید که می‌توان هر ماتریس سه بُعدی را (لااقل در حالت حدّی) محتوی نسخه‌های «یک» بلوک از تصویر با تحقّق‌های مختلفی از نویز در نظر گرفت. از طرفی، همانطور که از تئوری تخمین می‌دانیم، هرچه تعداد مشاهدات ما از یک داده‌ی نویزی بیش‌تر باشد، واریانس تخمین کم‌تر خواهد بود (این کمیّت با تعداد مشاهدات رابطه‌ی عکس دارد). بنابراین، نتیجه‌ی نویززدائی نسبت به حالتی که روی هر بلوک از تصویر به تنهائی کار می‌کنیم بهتر خواهد بود.

۴-۴ نویززدائی در دیکشنری‌های آموزش دیده

تصویرِ نویزیِ $\mathbf{y} \in \mathbb{R}^N$ را (که دارای $\sqrt{N} \times \sqrt{N}$ پیکسل است) در نظر بگیرید. بنا به دلایلی که در مقدمه گفتیم، این تصویر را به تعدادی تصویر کوچک‌تر هر یک به ابعاد $\sqrt{n} \times \sqrt{n}$ تقسیم کرده، و نهایتاً هر کدام را به یک بُردارِ معادلِ به طول n تبدیل می‌کنیم. برای بلوک شماره‌ی k داریم $\mathbf{p}_k = \mathbf{R}_k\mathbf{y}$ که در آن $\mathbf{R}_k \in \mathbb{R}^{n\times N}$ عملگری است

[۱] Block-Matching 3D Filtering

که این بلوک از تصویر را استخراج می‌کند. ترانهاده‌ی[۱] این ماتریس عکس این عمل را انجام می‌دهد. بعبارت دیگر، $\mathbf{R}^T\mathbf{p}_k$ تصویری (یا بعبارت بهتر، بُرداری) است که به جز قسمتِ متناظر با $\mathbf{p}_k$، بقیه‌ی آن صفر است. نهایتاً برای تخمین MAP تصویر تمیز داریم:

$$\left\{\{\hat{\mathbf{q}}_k\}_{k=1}^{M}, \hat{\mathbf{y}}\right\} = \arg\min_{\mathbf{z},\{\mathbf{q}_k\}} \left\{\lambda\|\mathbf{z}-\mathbf{y}\|_2^2 + \sum_{k=1}^{M}\mu_k\|\mathbf{q}_k\|_0 + \sum_{k=1}^{M}\|\mathbf{D}\mathbf{q}_k - \mathbf{R}_k\mathbf{z}\|_2^2\right\} \qquad (۴\text{-}۴)$$

که M برابر است با تعداد کلّ بلوک‌های استخراج شده. جمله‌ی اول در تابع هدف فوق نزدیکیِ بین تصویر نویزی با نسخه‌ی بی‌نویز شده‌ی آن را اندازه می‌گیرد. جمله‌ی دوم و سوم بیانگر اطّلاعات پیشین تصویر بوده که بر این نکته استوار است که هر بلوک در دیکشنریِ $\mathbf{D}$ یک نمایش تُنُک با خطای محدود دارد؛ یعنی $\|\mathbf{D}\mathbf{q}_k - \mathbf{p}_k\|_2 \le \epsilon$، که ϵ رابطه‌ی مستقیمی با انحراف معیارِ نویز موجود در $\mathbf{p}_k$ دارد. بعبارت دیگر $\epsilon = c\sqrt{n}\sigma$ که c یک عدد ثابت است.

برای حل مسأله‌ی (۴-۴) از الگوریتم بهینه‌سازی نوبتی استفاده می‌شود؛ به این ترتیب که ابتدا قرار می‌دهیم $\mathbf{z} = \mathbf{y}$ و سپس مسأله‌ی حاصل را که معادل است با بدست آوردن نمایش تُنُک همه‌ی بلوک‌ها، حل می‌کنیم. برای این منظور از الگوریتم OMP استفاده می‌شود که در آن برای بدست آوردن نمایش تُنُکِ $\mathbf{p}_k$ در دیکشنریِ $\mathbf{D}$ (که آن را با $\hat{\mathbf{q}}_k$ نشان می‌دهیم) آنقدر اتم انتخاب می‌کنیم تا خطای نمایش به زیرِ ϵ برسد. بعد از انجام این کار برای همه‌ی بلوک‌ها، نهایتاً مسأله‌ی (۴-۴) نسبت به $\mathbf{z}$ حل می‌شود. جواب این مسأله براحتی بصورت زیر بدست می‌آید:

$$\hat{\mathbf{y}}_0 = \left(\lambda\mathbf{I} + \sum_{k=1}^{M}\mathbf{R}_k^T\mathbf{R}_k\right)^{-1}\left(\lambda\mathbf{y} + \sum_{k=1}^{M}\mathbf{R}_k^T\mathbf{D}\hat{\mathbf{q}}_k\right). \qquad (۵\text{-}۴)$$

رابطه‌ی فوق چیزی نیست جز متوسط‌گیری بلوک‌های بی‌نویز شده به همراه ضریبی از تصویر نویزی. تا این‌جا فرض کردیم که دیکشنریِ $\mathbf{D}$ معلوم است. برای گرفتن نتایج بهتر، این دیکشنری را می‌توانیم آموزش دهیم. برای این منظور، یک گزینه برای داده‌های آموزشی استفاده از بلوک‌های چندین تصویر تمیز است. امّا گزینه‌ی بهتر، استفاده از بلوک‌های خودِ تصویر نویزی است [۲۵]. برای این منظور، کافی است در مسأله‌ی (۴-۴)، $\mathbf{D}$ را نیز مجهول بگیریم. در اینصورت، برای حل کردن این مسأله بار دیگر از بهینه‌سازی نوبتی استفاده می‌کنیم؛ به این ترتیب که ابتدا یک دیکشنریِ اولیه (عموماً DCT فوق کامل) انتخاب کرده و قرار می‌دهیم $\mathbf{z} = \mathbf{y}$. سپس نمایش تُنُک بلوک‌ها را با قید انرژی خطای محدود بدست می‌آوریم. با استفاده از بلوک‌های بی‌نویز شده، تخمینِ تصویر تمیز را با استفاده از (۵-۴) بدست می‌آوریم. با این کار امّا سطح نویز موجود در تصویر عوض می‌شود و حتّی

[۱] Transpose

(احتمالاً) دیگر فرض گوسی بودن آن هم برقرار نخواهد بود. به همین دلیل، قبل از محاسبه‌ی تخمینِ تصویر تمیز، ابتدا چند تکرار بین به‌روز شدن دیکشنری و بدست آوردن نمایش تُنُک بلوک‌ها انجام شده و بعد از آن، تخمین نهائی تصویر تمیز محاسبه می‌شود. بعبارت بهتر، ابتدا با استفاده از یک الگوریتم آموزش دیکشنری (که در [۲۵] از K-SVD استفاده شده است)، یک دیکشنری از روی بلوک‌های تصویر نویزی (به عنوان داده‌های آموزشی) آموزش می‌دهیم. دقت کنید که در این مرحله بلوک‌ها بی‌نویز می‌شوند (بخش ۲-۴ را ببینید). در نهایت، با استفاده از بلوک‌های بی‌نویز شده و رابطه‌ی (۴-۵)، تخمین تصویر بدون نویز را بدست می‌آوریم.

۴-۵ جمع‌بندی

در این فصل، نویززدائی از تصاویر با استفاده از نمایش تُنُک را بررسی کردیم. ابتدا مسأله‌ی کلّی نویززدائی را مطرح کردیم و در ادامه مروری مختصر داشتیم بر تعدادی از رهیافت‌هائی که برای این منظور وجود دارد. در نهایت، نویززدائی از تصاویر با دیکشنریِ آموزش دیده را بررسی کردیم. تا این‌جا هرچه گفتیم مروری بود بر کارهای گذشته، که البته جمع‌آوری و دسته‌بندی آن به صورت ارائه شده از کارهای نگارنده بوده و می‌توان آن را نیز جزء دستاوردهای این پایان‌نامه به حساب آورد. امّا فصل‌هائی که در ادامه می‌آید تماماً کارهائی است که نگارنده پیشنهاد کرده است. در فصل بعد، چند الگوریتم پیشنهادی برای بازیابی نمایش تُنُک سیگنال‌ها را معرفی می‌کنیم. فصل بعد از آن هم شامل چندین الگوریتم پیشنهادی برای آموزش دیکشنری است.

فصل ۵

الگوریتم GIRLS برای کُدینگ تُنُک

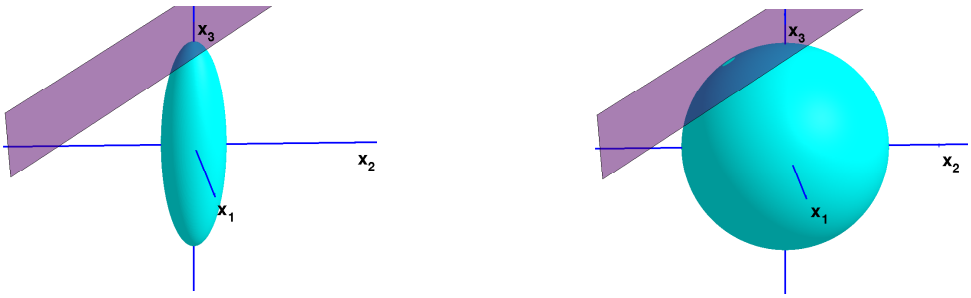
۵-۱ مقدمه

در فصل‌های گذشته، نمایش تُنُک و آموزش دیکشنری را به همراه تعدادی از الگوریتم‌های موجود مرور کردیم. این فصل و فصل بعدی به الگوریتم‌های پیشنهادی نگارنده اختصاص دارد. در فصل حاضر، الگوریتم GIRLS را به عنوان تعمیمی از الگوریتم‌های IRLS معرفی می‌کنیم. برای این منظور، ابتدا الگوریتم‌های IRLS را با جزئیات بیش‌تری بررسی کرده و تعابیر مختلفی از نحوه‌ی عملکرد این الگوریتم‌ها بیان می‌کنیم. سپس با الهام از این تعابیر، الگوریتم GIRLS[1] را معرفی می‌کنیم. در ادمه، یک تعبیر جالب از الگوریتم‌های IRLS از دیدگاه آماری و به نقل از مراجع مربوطه بیان می‌کنیم. این دیدگاه که مبتنی بر تئوری تخمین است دید بهتری از الگوریتم GIRLS ارائه می‌دهد. در انتها نیز عملکرد الگوریتم معرفی شده را با انجام شبیه‌سازی بررسی خواهیم کرد.

۵-۲ نگاهی دقیق‌تر به الگوریتم‌های IRLS

در فصل ۲ خانواده‌ی الگوریتم‌های IRLS را معرفی کردیم. در این بخش، با جزئیات بیش‌تری این الگوریتم‌ها را بررسی می‌کنیم. همانطور که در فصل ۲ گفتیم، این الگوریتم‌ها بصورت تکراری، رفتار نُرم ℓ_p را با استفاده از نُرم ℓ_2 وزن‌دهی شده تقریب می‌زنند. برای این منظور، تکرارهائی بصورت زیر انجام می‌شود:

[1] Generalized Iteratively Re-weighted Least Squares



(الف) $\mathbf{w} = [۱, ۱, ۱]$ (ب) $\mathbf{w} = [۱۰, ۱۰, ۱]$

شکل ۵-۱: سطوح ثابتِ $\mathbf{x}^T\mathbf{W}\mathbf{x}$ بازای دو حالت مختلف برای درایه‌های روی قطر اصلیِ $\mathbf{W}$ که با بُردار $\mathbf{w}$ نشان داده شده‌اند.

$$\mathbf{x}^{(k+۱)} = \arg\min_{\mathbf{x}} \sum_i w_i^{(k)} x_i^۲ = \mathbf{x}^T\mathbf{W}_k\mathbf{x} \quad \text{subject to} \quad \mathbf{y} = \mathbf{D}\mathbf{x}, \tag{۵-۱}$$

که در آن $\mathbf{W}_k = \text{diag}(w_i^{(k)} = |x_i^{(k)} + \sigma|^{p-۲})$. جواب این مسأله عبارت است از

$$\mathbf{x}^{(k+۱)} = \bar{\mathbf{W}}_k\mathbf{D}^T(\mathbf{D}\bar{\mathbf{W}}_k\mathbf{D}^T)^{-۱}\mathbf{y}. \tag{۵-۲}$$

که در آن $\bar{\mathbf{W}}_k \triangleq \mathbf{W}^\dagger{}_k$. در حالت کلّی امّا لزومی ندارد که در این‌جا خود را محدود به تابعِ نُرم ℓ_p کنیم؛ بلکه می‌توانیم هر تابع تشویق‌کننده تُنُک بودن را نیز با استفاده از ایده‌ی این الگوریتم‌ها با یک نُرم $\ell_۲$ وزن‌دهی شده تقریب بزنیم. برای توضیح بیش‌تر، ابتدا دقت کنید که با توجه به رابطه‌ی $\mathbf{x}^T\mathbf{W}\mathbf{x} = \sum_i w_i x_i^۲$، عناصر w_i باید به‌گونه‌ای باشند که تابع هدف جدید طی پیشروی الگوریتم، تُنُک بودن جواب را تضمین کند، یا به سمت جواب‌های تُنُک حرکت کند. برای این منظور، عنصر w_i باید برای مقادیر کوچک (از نظر قدرمطلق) x_i مقداری بزرگ داشته و بر عکس، برای مقادیر بزرگ x_i مقداری کوچک داشته باشد. به این ترتیب، درایه‌های کوچکِ بُردار $\mathbf{x}$ زیاد جریمه می‌شوند، و بر عکس، درایه‌های بزرگِ آن کمتر جریمه خواهند شد. بعبارت دیگر، اگر درایه‌ی x_i در یک تکرار خیلی کوچک باشد، وزن متناظر آن، یعنی w_i، در تکرار بعدی مقدار خیلی بزرگی خواهد داشت که این منجر به صفر شدن x_i در این تکرار می‌شود. به همین دلیل، وزن‌های اولیّه همگی برابر با «یک» انتخاب می‌شوند تا در تکرار نخست، همه‌ی درایه‌های $\mathbf{x}$ به یک اندازه جریمه شوند.

رفتار وزن‌های $\{w_i\}$ بصورتی که در پاراگراف قبل ذکر شد یک تعبیر هندسی جالب دارد. در تکرار نخست، همانطور که گفتیم، این وزن‌ها همگی برابر با «یک» انتخاب می‌شوند. با این انتخاب، شکل سطوحِ ثابتِ تابع هدف، یعنی سطوحِ «ثابت$= \mathbf{x}^T\mathbf{W}\mathbf{x}$» کُروی شکل بوده و در واقع جواب تکرار بعدی که از مسأله‌ی (۵-۱) بدست می‌آید چیزی نیست جز جوابِ با حدّاقل نُرم $\ell_۲$ دستگاه معادلات $\mathbf{y} = \mathbf{D}\mathbf{x}$. در تکرارهای بعدی، وزن‌های $\{w_i\}$ برحسب

عناصر بُردار جواب مرحله قبل تغییر می‌کنند و شکل سطوح ثابت تابع هدف به صورت بیضی‌هائی (یا به عبارت دقیق‌تر، بیضیگون‌هائی) به مرکز مبدأ و با قطرهائی موازی محورهای مختصات تبدیل می‌شود. در واقع هر چه تعداد تکرار بالا می‌رود، کشیدگی این بیضی‌ها در جهت عناصر با قدرمطلق بزرگِ بُردار جواب قبلی بیش‌تر می‌شود و به این ترتیب به سمت جواب‌های تُنُک پیش می‌رویم. شکل ۵-۱ دو مورد از این سطوح را نشان می‌دهد. در این شکل، بُردار $\mathbf{w}$ نشان‌دهنده‌ی عناصر روی قطر اصلیِ ماتریس $\mathbf{W}$ است، یعنی $\mathbf{W} = \mathrm{diag}(\mathbf{w})$.

بعنوان یک تعبیر دیگر از نحوه‌ی عملکرد الگوریتم‌های IRLS، دقت کنید که چون درایه‌های روی قطر ماتریس $\mathbf{W}_k$ همگی نامنفی‌اند، بنابراین این ماتریس را می‌توان برحسب ریشه‌ی دوم آن بصورت زیر تجزیه کرد:

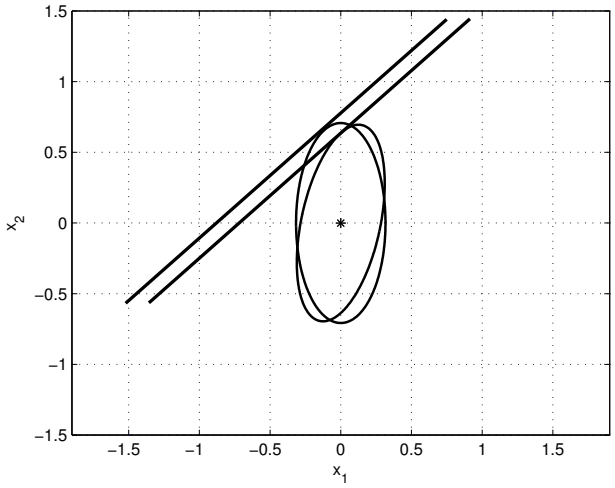
$$\mathbf{W}_k = \mathbf{\Omega}_k \mathbf{\Omega}_k, \tag{۵-۳}$$

که در آن $\mathbf{\Omega}_k = \mathbf{W}_k^{۱/۲}$. با یک تغییر متغیّر ساده، مسأله‌ی (۵-۱) را براحتی می‌توان به مسأله‌ی زیر تبدیل کرد:

$$\mathbf{x}^{(k+۱)} = \mathbf{\Gamma}_k . \left\{ \arg\min_{\mathbf{z}} \|\mathbf{z}\|_۲^۲ \quad \text{subject to} \quad \mathbf{y} = \mathbf{C}_k \mathbf{z} \right\}, \tag{۵-۴}$$

که در آن $\mathbf{\Gamma}_k = \mathbf{\Omega}_k^{-۱}$ و $\mathbf{C}_k = \mathbf{D}\mathbf{\Gamma}_k$. مسأله‌ی مینیمم‌سازی برحسب متغیّر $\mathbf{z}$ مشابه تکرارِ اوّل الگوریتم‌های IRLS، عبارتست از پیدا کردن جواب حداقل نُرمِ $\ell_۲$ دستگاه معادلات $\mathbf{y} = \mathbf{C}_k \mathbf{z}$ که در آن، ماتریس $\mathbf{C}_k$ همان ماتریس $\mathbf{D}$ است با این تفاوت که در هر تکرار، ستون‌های آن متناسبِ با قدرمطلق عناصر متناظر در بُردار جواب قبلی، $\mathbf{x}^{(k)}$، وزن داده می‌شود. حال اگر عنصری از بُردار جواب قبلی برابر صفر بوده یا مقداری خیلی کوچک داشته باشد، ستون متناظر آن در ماتریس $\mathbf{D}$ با ضرب شدن در این مقدار کوچک تقریباً با بُردار صفر جایگزین شده و در نتیجه، در ماتریس $\mathbf{C}_k$ این ستون صفر می‌شود. بنابراین، با پیشروی الگوریتم، ستون‌های ماتریس $\mathbf{C}_k$ تبدیل به بُردار صفر می‌شوند، به جز ستون‌هائی که جواب واقعی تُنُک مسأله (البته در صورت همگرائی الگوریتم به این جواب) از آن‌ها استفاده کرده است. با این توضیحات، این ایده به ذهن می‌رسد که برای بدست آوردن جوابِ تُنُک دستگاه معادلات $\mathbf{y} = \mathbf{D}\mathbf{x}$، بصورت تکراری جواب‌های حداقل نُرم $\ell_۲$ این دستگاه را بدست آورده و برای تکرار بعدی، ستون‌هائی از ماتریس $\mathbf{D}$ را که متناظر با مقادیر خیلی کوچکِ عناصر بُردار جواب فعلی است، حذف کنیم. به این ترتیب، سرعت همگرائی الگوریتم سریع‌تر می‌شود. البته مشکل این روش پیدا کردن یک آستانه‌ی مناسب برای حذف ستون‌های ماتریس $\mathbf{D}$ است؛ کاری که الگوریتم‌های IRLS معمولی به صورت وفقی و هوشمند انجام می‌دهند.

بعنوان جایگزینی دیگر، می‌توانیم وزن‌های $\{w_i\}$ را در مسأله‌ی (۵-۱) بصورت زیر انتخاب کنیم:



شکل ۵-۲: شکل سطوح ثابت به همراه قید مسأله برای دو حالت $\mathbf{W}$ قطری و غیر قطری.

$$w_i = \frac{۱}{\alpha^{x_i^{(k)}} - ۱}, \tag{۵-۵}$$

که در آن α یک پارامتر ثابت است. دقت کنید که این انتخاب با آنچه که در ابتدای این بخش گفتیم هم‌خوانی دارد؛ یعنی برای یک x_i کوچک، مقدار w_i بزرگ خواهد بود و برعکس. α یک عدد ثابت بزرگتر از یک است و طبیعتاً هرچه مقدار آن بزرگتر باشد، مقدار جریمه‌ی متناظر برای درایه‌های کوچکِ $\mathbf{x}$ شدیدتر بوده و برای درایه‌های بزرگِ آن خفیف‌تر خواهد بود.

۵-۳ الگوریتم IRLS تعمیم‌یافته

در این بخش، الگوریتم GIRLS را معرفی می‌کنیم. ابتدا توجه کنید که حالت کلّی‌تر ماتریس وزن‌ها، یعنی $\mathbf{W}_k$، عبارت است از یک ماتریس متقارن مثبت-معیّن[۱] (نه لزوماً قطری). در این حالت، سطوح ثابتِ تابع هدفِ مسأله‌ی (۵-۱) بیضیگون‌هائی به مرکز مبدأ بوده که نسبت به محورهای مختصات دَوَران پیدا کرده‌اند[۲]. همانطور که پیش‌تر اشاره شد، ماتریس $\mathbf{W}_k$ باید به گونه‌ای باشد که الگوریتم را در هر تکرار به سمت جواب‌های تُنُک پیش ببرد. برای حالت قطریِ این ماتریس دیدیم که اگر درایه‌های روی قطر آن درست انتخاب شود این وضعیت رخ می‌دهد. اما برای حالت کلّی این ماتریس، درایه‌های آن چگونه باید انتخاب شوند؟ و آیا اصلاً در حالت غیرقطری به جواب‌های تُنُک می‌رسیم یا خیر؟ در پاسخ باید گفت که اگر درایه‌های این ماتریس به درستی انتخاب شود، همگرائی الگوریتم

[۱] Positive-definite

[۲] البته حالت کلّی تابع هدف بصورت $(\mathbf{x}-\mathbf{x}_۰)^T\mathbf{W}_k(\mathbf{x}-\mathbf{x}_۰)$ است که بُردار $\mathbf{x}_۰$ مرکز بیضیگون است. ما در اینجا مرکز بیضیگون‌ها را مبدأ مختصات گرفته‌ایم.

و کیفیت جواب نهائی نسبت به حالتِ با ماتریس وزنیِ قطری می‌تواند بهتر شود. شکل ۵-۲ وضعیت سطوح ثابت را برای دو حالت ماتریس قطری و ماتریس متقارن مثبت-معیّن کلّی برای یک مسأله‌ی ساده نشان می‌دهد. قید مسأله هم بصورت یک خط راست رسم شده است. همانطور که در این شکل دیده می‌شود، با انتخاب درست درایه‌های ماتریس کلّی، همگرائی الگوریتم نسبت به حالت قطری سریع‌تر خواهد شد.

اگر تجزیه‌ی مقادیر ویژه‌ی ماتریس $\mathbf{W}$ را به صورت $\mathbf{W}=\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$ بنویسم، به راحتی می‌توان نشان داد که مسأله‌ی (۵-۱) معادل با مسأله‌ی زیر است[۱]:

$$\mathbf{x}=\mathbf{Q}.\left\{\arg\min_{\mathbf{z}}\ \mathbf{z}^T\mathbf{\Lambda}\mathbf{z}\quad \text{subject to}\quad \mathbf{y}=\mathbf{C}\mathbf{z}\right\}, \tag{۵-۶}$$

که در آن $\mathbf{C}=\mathbf{D}\mathbf{Q}$. این فُرم مسأله در واقع ابتدا یک مسأله‌ی IRLS معمولی را حل می‌کند (که البته ماتریس $\mathbf{D}$ در آن به ماتریس $\mathbf{C}$ تبدیل شده است)، سپس جواب نهائی با چرخاندن این جواب با ماتریس دوران $\mathbf{Q}$ بدست می‌آید. با یک دید شهودی، ماتریس $\mathbf{\Lambda}$ را می توان بصورت $\mathbf{\Lambda}=\mathrm{diag}(|z_i^{(k)}+\sigma|^{p-۲})$ انتخاب کرد. تنها مجهول این مسأله ماتریس $\mathbf{Q}$ است. اگر بتوان این ماتریس (اُرتونرمال) را طی یک مسأله بهینه‌سازی بصورت سراسری یا در هر تکرار تعیین کرد به گونه‌ای که جواب نهائی مسأله تُنُک باشد، می توان ادّعا کرد که حالت کلّی الگوریتم‌های IRLS را بدست آورده‌ایم. با این حال، یک حالت (زیربهینه) از ماتریس وزنی را در ادامه بررسی می‌کنیم.

در الگوریتم IRLS کلّی معرفی شده، یک گزینه برای ماتریس $\bar{\mathbf{W}}_k$ به صورت زیر است:

$$\bar{\mathbf{W}}_k=|\mathbf{x}_k\mathbf{x}_k^T|. \tag{۵-۷}$$

یک مشکل اساسی که این انتخاب دارد این است که در رابطه‌ی (۵-۲) جمله‌ی $(\mathbf{D}\bar{\mathbf{W}}_k\mathbf{D}^T)$ رُتبه-یک و در نتیجه معکوس‌ناپذیر می‌شود (این موضوع به سادگی قابل اثبات است). برای رفع این مشکل انتخاب زیر را پیشنهاد می‌کنیم:

$$\bar{\mathbf{W}}_k=\mathbf{W}_k^d+\mathbf{W}_k^{off},\quad \mathbf{W}_k^d=\mathrm{diag}(\alpha^{|x_i^{(k)}|}-۱),\quad \mathbf{W}_k^{off}=\mathrm{Thresh}(|\mathbf{x}_k\mathbf{x}_k^T|)_{\mathrm{diag}=۰} \tag{۵-۸}$$

در این رابطه ماتریس $\mathbf{W}_k^{off}$ همان رابطه (۵-۷) بوده با این تفاوت که همه درایه‌های روی قطر اصلی آن و نیز درایه‌های خارج از قطر آن که مقدارشان زیر یک آستانه است، صفر شده است. این آستانه بصورت کسری از بزرگترین درایه‌ی $|\mathbf{x}_k\mathbf{x}_k^T|$ انتخاب می‌شود. در ادامه، توجیهی برای این کار بیان می‌کنیم.

دلیل انتخاب درایه های روی قطر اصلی ماتریس $\mathbf{W}_k^{off}$ به صورت معادله (۵-۵) این است که مشکل

[۱] برای راحتی نماد k یا همان شماره تکرار را حذف کرده‌ایم.

رُتبه-یک بودن و در نتیجه معکوس ناپذیری برطرف می‌شود. انتخاب درایه‌های خارج قطر اصلی آن طبق رابطه ۵-۸ یک توجیه شهودی به این ترتیب دارد که ابتدا توجه کنید که در معادله‌ی بیضی دوران یافته به صورت زیر:

$$\mathbf{x}^T\mathbf{W}\mathbf{x} = \sum_i w_{ii}x_i^2 + \sum_{i\neq j} w_{ij}x_ix_j, \tag{۹-۵}$$

اگر ضریب w_{ij} صفر باشد، به این معنی است که (تصویر) بیضیگون روی مؤلفه‌های x_i و x_j هیچ دورانی ندارد؛ بنابراین اگر این ضرایب در جواب تُنُک واقعی مسأله هم صفر باشند، به طور شهودی نباید در زیرفضای شامل این دو مؤلفه دورانی داشته باشیم. لذا بهترین حالت این است که در هر تکرارِ الگوریتم، فقط تعداد معیّنی از این حاصل‌ضرب‌ها را نگه داشته و بقیه را صفر کنیم.

اگر تعداد عناصر غیر صفر جواب واقعی مسأله برابر با s باشد، تعداد حاصل‌ضرب‌هائی که در هر تکرار باید نگه داریم برابر است با $s \times (s-1)/2$. اما از آن‌جا که در برخی کاربردها این تعداد را از قبل نمی‌دانیم، باید یک آستانه‌ی مناسب برای الگوریتم انتخاب کنیم (حالتی را که در آن از تعداد عناصر غیر صفر بُردار جواب واقعی برای حل الگوریتم استفاده کرده‌ایم، O-GIRLS[1] نام نهاده‌ایم؛ به این دلیل که در آن از اطّلاعات اضافی برای حل مسأله استفاده شده است).

ممکن است این ایراد گرفته شود که ما روی درایه‌های معکوس ماتریس وزنی آستانه‌گذاری می‌کنیم؛ در صورتی که طبق توجیه ذکر شده، این کار تنها روی خود ماتریس وزنی، یعنی $\mathbf{W}_k$، معنی دارد. در پاسخ باید گفت با توجه به این که این ماتریس در هر تکرار خیلی تُنُک است، موقعیت درایه‌های صفر آن در ماتریس معکوسش حفظ می‌شود و چیزی که اهمیت دارد موقعیت این درایه‌ها است.

۵-۴ نگاه آماری به GIRLS

یک تعبیر جالب از الگوریتم‌های IRLS با استفاده از دیدگاه تئوری تخمین وجود دارد [۶۰، ۶۳]. در این رهیافت، بُردار مشاهدات بصورت $\mathbf{y} = \mathbf{D}\mathbf{x} + \mathbf{n}$، که $\mathbf{n} \sim \mathcal{N}(0, \sigma^2\mathbf{I})$ یک نویز گوسی سفید است، مدل می‌شود. سپس با استفاده از تخمین MAP، بُردار $\mathbf{x}$ تخمین زده می‌شود. توزیع پیشینی که برای هر یک از درایه‌های $\mathbf{x}$ فرض می‌شود، گوسی با میانگین صفر و یک واریانس مشخص است و بعلاوه، این درایه‌ها مستقل خطی فرض می‌شوند. بعبارت دیگر، توزیع بُردار $\mathbf{x}$ بصورت زیر است:

[1] Oracle GIRLS

$$p(\mathbf{x};\mathbf{W}) \propto \sqrt{|\mathbf{W}|}\exp(-\frac{۱}{۲}\mathbf{x}^T\mathbf{W}\mathbf{x}), \tag{۵–۱۰}$$

که در آن $\mathbf{W}=\mathrm{diag}(w_i=۱/\sigma_i^۲)$ و $\sigma_i^۲$ واریانس x_i است. دقت کنید که $\sigma_i^۲$ ها از قبل معلوم نیستند و طی فرآیند حلّ مسأله تخمین زده می‌شوند. استفاده از تخمین MAP نهایتاً منجر به مسأله‌ی زیر می‌شود [۶۳]:

$$\min_{\mathbf{x},\{w_i\}} \mathbf{x}^T\mathbf{W}\mathbf{x} - \ln|\mathbf{W}| + \frac{۱}{\sigma^۲}\|\mathbf{y}-\mathbf{D}\mathbf{x}\|_۲^۲. \tag{۵–۱۱}$$

حلّ این مسأله با استفاده از مینیمم‌سازی نوبتی روی $\mathbf{x}$ و w_i ها منجر به تکرارهای زیر می‌شود:

$$\begin{cases} \mathbf{x} \longleftarrow (\mathbf{D}^T\mathbf{D}+\sigma^۲\mathbf{W})^{-۱}\mathbf{D}^T\mathbf{y} \\ \forall i:\ w_i \longleftarrow \frac{۱}{\sigma_i^۲} = \frac{۱}{|x_i|^۲}. \end{cases} \tag{۵–۱۲}$$

در [۶۳] نشان داده شده است که تکرارهای فوق زمانیکه $\sigma^۲ \to ۰$ منجر به همان الگوریتم IRLS معمولی (حالت بدون نویز) می‌شود.

با توضیحات فوق، می‌توان گفت که از دیدگاه آماری، در الگوریتم GIRLS برخلاف الگوریتم‌های IRLS، درایه‌های $\mathbf{x}$ وابسته‌ی خطی فرض شده و در نتیجه ماتریس کواریانس در حالت کلّی غیرقطری خواهد بود. فرض استقلال آماری ضرایب نمایش تُنُک به نوعی به این معنی است که انتخاب هر اتم برای نمایش یک سیگنال کاملاً مستقل از دیگری است. این فرض امّا در عمل و بخصوص در مورد سیگنال‌های طبیعی فرض مناسبی نیست. این موضوع در [۳۰] به تفصیل مورد بحث قرار گرفته است. بعنوان مثال، ضرایب بزرگ در نمایش یک بلوک از تصویر با استفاده از دیکشنری DCT، بیش‌تر مربوط به اتم‌های با فرکانس پائین هستند. بعبارت دیگر، علاوه بر مقدار ضرایب، «الگوی» ضرایب غیرصفر نیز باید مورد توجه قرار بگیرد. همانطوری که در [۳۰] بررسی شده است، بکارگیری فرض وابستگی بین ضرایب نمایش تُنُک منجر به بهبود کیفیت جواب‌های کُدینگ تُنُک خواهد شد. حال با این توضیحات، GIRLS را می‌توان الگوریتمی دانست در جهت میل به این هدف. در ادامه، این الگوریتم را از این دیدگاه بررسی می‌کنیم.

قبل از هر چیز، ابتدا مفهوم «ردّ»[۱] یک ماتریس را یادآوری می‌کنیم. ردّ ماتریس $\mathbf{A}$ بصورت $\mathrm{Tr}(\mathbf{A}) \triangleq \sum_i a_{ii}$ تعریف می‌شود. با این توضیح، برای تعیین وزن‌های بهینه کافی است مشتق تابع هدف مسأله‌ی (۵–۱۱) نسبت به w_{ij} را برابر با صفر قرار دهیم. در اینصورت با توجه به این که $\partial \ln|\mathbf{A}| = \mathrm{Tr}(\mathbf{A}^{-۱}\partial\mathbf{A})$، بدست می‌آوریم:

$$x_i x_j = \mathrm{Tr}(\mathbf{W}^{-۱}\hat{\mathbf{W}}) = \bar{w}_{ij}, \tag{۵–۱۳}$$

که در آن $\hat{\mathbf{W}}$ ماتریسی است که غیر از درایه‌ی موجود در موقعیت (i,j) که برابر با ۱ است، بقیه‌ی درایه‌های آن

[۱] Trace

همگی برابر با صفراند. $\bar{w}_{ij}$ هم درایه‌ی (i,j)اُم ماتریس $\mathbf{W}^{-1}$ است. به این ترتیب بدست می‌آوریم $\mathbf{W}^{-1} = \mathbf{x}\mathbf{x}^T$. در نتیجه، چون این ماتریس رُتبه-۱ است، بنابراین برای بدست آوردن ماتریس $\mathbf{W}$ به مشکل برمی‌خوریم که البته با توضیحاتی که در قسمت قبل ارائه دادیم، می‌توانیم از تقریب $w_{ij} = ۱/(x_i x_j)$ استفاده کنیم.

۵-۵ شبیه‌سازی

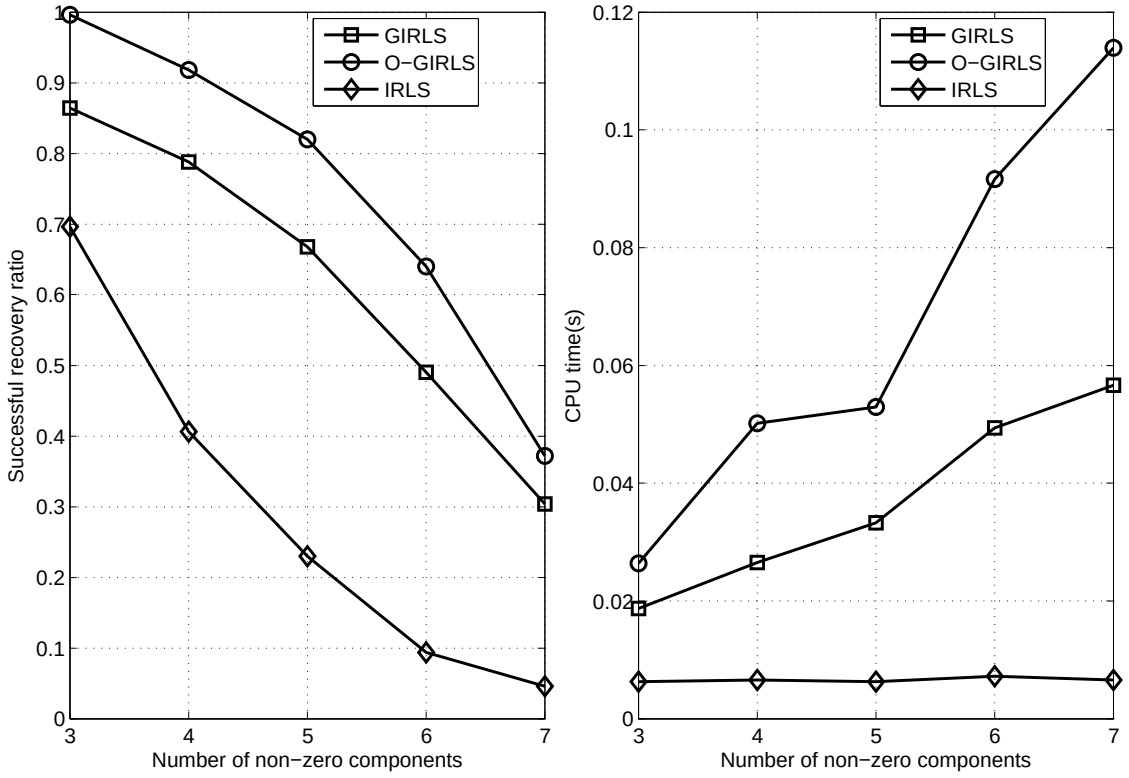
در این بخش عملکرد سه الگوریتم GIRLS، O-GIRLS و IRLS معمولی (با انتخاب $p = ۰$) را با هم مقایسه می‌کنیم. برای این منظور، ابتدا یک دیکشنری با ابعاد ۱۰۰ × ۲۰ می‌سازیم. درایه‌های این دیکشنری را از توزیع گوسی با میانگین صفر و واریانس یک انتخاب کرده و در آخر ستون‌های آن را نُرمالیزه می‌کنیم. سپس بُردار مشاهدات را بصورت $\mathbf{y} = \mathbf{D}\mathbf{x}$ با استفاده از ترکیب خطی s ستون از این دیکشنری تشکیل می‌دهیم. درایه‌های غیر صفرِ $\mathbf{x}$ مشابه درایه‌های دیکشنری و موقعیت آن‌ها نیز بصورت تصادفی با توزیع یکنواخت انتخاب می‌شود. s را از ۳ تا ۷ تغییر می‌دهیم. برای ارزیابی عملکرد سه الگوریتم مذکور، بُردار $\mathbf{y}$ را به همراه دیکشنری $\mathbf{D}$ به این الگوریتم‌ها داده و نتیجه‌ی خروجی آن‌ها را با مقدار واقعی $\mathbf{x}$ مقایسه می‌کنیم. مشابه برخی از مراجع، می‌گوئیم یک الگوریتم با موفقیت بُردار $\mathbf{x}$ را یافته است اگر

$$\|\mathbf{x} - \hat{\mathbf{x}}\|_\infty \leq ۱۰^{-۳}, \tag{۱۴-۵}$$

که $\hat{\mathbf{x}}$ خروجی الگوریتم است. $\|.\|_\infty$ نُرم ℓ_∞ بوده که بصورت $\|\mathbf{x}\|_\infty = \max_i |x_i|$ تعریف می‌شود. هر آزمایش را ۵۰۰ بار تکرار کرده و در انتها نتایج را بصورت درصد موفقیت گزارش می‌دهیم. لازم به ذکر است که در این‌جا هدف نمایش دقیق سیگنال بوده و بنابراین نویز برابر با صفر فرض می‌شود. برای بررسی مقاومت این الگوریتم‌ها نسبت به نویز باید نسخه‌ی پایدارشان را که بصورت زیر است حل کنیم، که البته ما این را جزء کارهای آینده قرار می‌دهیم.

$$\mathbf{x}^{(k+۱)} = \arg\min_{\mathbf{x}} \mathbf{x}^T \mathbf{W}_k \mathbf{x} + \lambda \|\mathbf{y} - \mathbf{D}\mathbf{x}\|_۲^۲. \tag{۱۵-۵}$$

این آزمایش‌ها با سیستمی دارای پردازنده‌ی Intel Core i3، سرعت پردازش ۲٫۱۳ GHz، حافظه‌ی ۲ GB و تحت نرم‌افزار MATLAB نسخه‌ی R2011a انجام شده است. نمودار درصد موفقیت و زمان متوسط صرف شده توسط هر یک از این الگوریتم‌ها در شکل ۵-۳ آورده شده است. با دقت در این شکل‌ها می‌توان گفت که درصد موفقیت الگوریتم GIRLS نسبت به الگوریتم IRLS معمولی بهتر است. هم‌چنین، الگوریتم O-GIRLS نسبت به GIRLS



شکل ۵–۳: نمودارهای درصد موفقیت و زمان متوسط صرف شده توسط سه الگوریتم GIRLS، O-GIRLS و IRLS برای مسأله‌ای با ابعاد $n = ۲۰$ و $m = ۱۰۰$.

از این نظر عملکرد بهتری داشته است. از نظر زمان صرف شده امّا الگوریتم IRLS معمولی نسبت به دو الگوریتم دیگر وضعیت بهتری دارد. برای این الگوریتم زمان سپری شده تقریباً مستقل از تعداد درایه‌های غیر صفر جواب است.

از دیدگاه حسگری فشرده، در این آزمایش ما می‌خواهیم سیگنالی با بُعد ۱۰۰ را که در یک پایه‌ی مشخص نمایشی با تنها s درایه‌ی غیرصفر دارد، با استفاده از ۲۰ اندازه‌گیری تصادفی از آن بازیابی کنیم. همانطور که از نتایج مشخص است، زمانی که تعداد درایه‌های غیرصفر نمایش تُنُک سیگنال برابر با ۳ است، الگوریتم O-GIRLS با موفقیت نمایش سیگنالِ ۱۰۰ بُعدی مورد نظر را بازیابی می‌کند.

۵-۶ جمع‌بندی

در این بخش ابتدا الگوریتم‌های IRLS را با جزئیات بیش‌تری بررسی کرده و تعبیرهای مختلفی از این الگوریتم‌ها ارائه کردیم. دیدیم که در هر تکرار از این الگوریتم‌ها، سطوح ثابت طوری تغییر شکل پیدا می‌کنند که به سمت جواب‌های تُنُک پیش برویم. این تغییر شکل عبارتست از تغییر طول قطرهای بضیگون‌ها به گونه‌ای که اگر ضریبی در یک تکرار اندازه‌ی نسبتاً بزرگی داشته باشد، طول قطر متناظر با آن برای تکرار بعدی افزایش خواهد یافت. با یک دید شهودی توضیح دادیم که اگر علاوه بر افزایش طول قطرها، بیضیگون‌ها در هر تکرار دوران هم پیدا کنند همگرائی الگوریتم سریع‌تر خواهد شد. به همین دلیل، الگوریتم GIRLS را به عنوان تعمیمی از الگوریتم‌های IRLS معمولی ارائه کردیم. به عنوان یک حالت زیربهینه، یک گزینه برای ماتریس وزن‌ها پیشنهاد کردیم. در ادامه، الگوریتم‌های IRLS و GIRLS را از دیدگاه تئوری تخمین بررسی کردیم. این دیدگاه دید شهودی‌تری نسبت به الگوریتم GIRLS به ما می‌دهد که عبارتست از بکارگیری وابستگی بین اتم‌ها برای بازیابی نمایش تُنُک. این دیدگاه امّا نیازمند بررسی بیش‌تری است و بنابراین ما آن را به کارهای آینده موکول می‌کنیم. در انتها با انجام یک شبیه‌سازی، عملکرد این الگوریتم‌ها را ارزیابی کردیم. مشکلی که الگوریتم GIRLS دارد زمان اجرای نسبتاً بالای آن است. بنابراین، یکی از جهت‌های کارهای آینده می‌تواند بهبود سرعت و عملکرد این الگوریتم باشد.

فصل ۶

الگوریتم‌های پیشنهادی برای آموزش دیکشنری

۶-۱ مقدمه

در فصل ۳ بحث «آموزش دیکشنری» برای پردازش تُنُک سیگنال‌ها را به همراه تعدادی از الگوریتم‌های موجود بررسی کردیم. اهمیت یک دیکشنری مناسب یا اصطلاحاً «تُنُک کننده» را نیز بیان کردیم. گفتیم که چنین دیکشنری‌ای این توانائی را دارد که برجسته‌ترین ویژگی‌های یک کلاس مشخص از داده‌ها را استخراج می‌کند. در این فصل چند الگوریتم جدید برای آموزش دیکشنری پیشنهاد می‌کنیم. این الگوریتم‌ها در دو دسته‌ی کلّی قرار دارند. الگوریتم‌های دسته‌ی اول مشابه K-SVD اتم‌ها را «یکی یکی» به‌روز می‌کنند، در حالی که الگوریتم‌های دسته‌ی دوم مشابه MOD همه‌ی اتم‌ها را «یکجا» به‌روز می‌کنند. با انجام شبیه‌سازی‌های متعدد، هم بر روی داده‌های مصنوعی و هم در کاربرد واقعی، عملکرد این الگوریتم‌ها را ارزیابی خواهیم کرد.

در ادامه‌ی این فصل، منظور از $\mathbf{X}(i,:)$ و $\mathbf{X}(:,j)$ به ترتیب، سطر i اُم و ستون j اُم ماتریس $\mathbf{X}$ است. $\mathbf{X}(\omega,j)$ نشان‌دهنده‌ی بُرداری ستونی شامل درایه‌هائی از ستون j اُم ماتریس $\mathbf{X}$ است که اندیس آن‌ها در ω وجود دارد. هم‌چنین، منظور از $SA(\mathbf{Y},\mathbf{D})$ بدست آوردن تقریب تُنُک سیگنال‌های $\mathbf{Y}$ در دیکشنریِ $\mathbf{D}$ است.

۶-۲ الگوریتم DL1

در فصل ۳ دیدیم که اکثر الگوریتم‌های آموزش دیکشنری شامل دو گام هستند. در گام اول با استفاده از دیکشنری فعلی، نمایش تُنُک همه‌ی داده‌های آموزشی محاسبه می‌شود. در این گام، محاسبه‌ی نمایش تُنُک داده‌ها یا بر اساس قیدِ حداکثر تعداد اتم‌های مجاز در نمایش داده‌ها است، یا رسیدن انرژی خطای تقریب به زیر یک آستانه‌ی مشخص. در گام دوم، دیکشنری طوری به‌روز می‌شود که خطای تقریبِ تُنُکِ داده‌ها در گام قبل حداقل شده یا کاهش یابد. گفتیم که تفاوت الگوریتم‌های آموزش دیکشنری عمدتاً در نحوه‌ی انجام این گام، یعنی گام «به‌روز کردن دیکشنری» است.

دو الگوریتم K-SVD و MOD را بررسی کردیم. یک تفاوت آشکار این دو الگوریتم در گام به‌روز کردن دیکشنری این است که K-SVD هر اتم را جداگانه (با ثابت نگه داشتن سایر اتم‌ها) به‌روز می‌کند، حال آنکه MOD همه‌ی اتم‌ها را «یکجا» به‌روز می‌کند. تفاوت دیگر این دو الگوریتم در این است که در K-SVD علاوه بر هر اتم، سطر متناظر آن در ماتریس ضرایب نیز به‌روز می‌شود؛ کاری که در MOD انجام نمی‌شود. این موضوع در حقیقت ضعف اساسی MOD نسبت به K-SVD است. برای توضیح بیش‌تر، خوب است ابتدا مسأله‌ی مینیمم‌سازی خطای تقریب در الگوریتم MOD را در این فصل تکرار کنیم[۱]:

$$\min_{\mathbf{D}} \|\mathbf{Y}-\mathbf{D}\mathbf{X}\|_F^2. \qquad (۱-۶)$$

ماتریس ضرایبِ $\mathbf{X}$ مربوط به گام اول الگوریتم است و بنابراین در مسأله‌ی فوق ثابت است. ثابت بودنِ $\mathbf{X}$ در این گام قید نسبتاً سنگینی برای به‌روز کردن دیکشنری است؛ چرا که در این حالت، درایه‌های $\mathbf{X}$ باید دقیقاً همان مقادیر گام قبلی را داشته باشند. در الگوریتم پیشنهادی شماره‌ی ۱، ما این قید را برمی‌داریم و در عوض، مشابه K-SVD تنها این قید را اعمال می‌کنیم که «ساپورتِ» ماتریس ضرایب یا همان موقعیت درایه‌ها، و نه «مقدار» آنها، ثابت است. ساپورت ماتریس $\mathbf{X}$ را با نماد $\text{supp}(\mathbf{X})$ نشان داده و بصورت $\text{supp}(\mathbf{X}) \triangleq \{(i,j) : \ x_{ij} \neq 0\}$ تعریف می‌کنیم. در نتیجه، در گام به‌روز کردن دیکشنری، درایه‌های غیرصفر ماتریس $\mathbf{X}$ از گام قبل را نیز به‌روز کرده و به عبارت دیگر مسأله‌ی زیر را در این گام حل می‌کنیم:

$$\left\{\mathbf{D}^{(k+1)}, \mathbf{X}^{(k+\frac{1}{2})}\right\} = \arg\min_{\mathbf{D},\mathbf{X}} \|\mathbf{Y}-\mathbf{D}\mathbf{X}\|_F^2 \quad \text{subject to} \quad \text{supp}(\mathbf{X}) = \text{supp}(\mathbf{X}^{(k)}). \qquad (۲-۶)$$

[۱] توجه کنید که قید نُرمالیزه بودن ستون‌های دیکشنری در تمام مسائل این فصل بطور ضمنی وجود دارد؛ اگرچه در برخی موارد بطور صریح ذکر نمی‌شود.

k نشان‌دهنده‌ی تکرار بین دو گام است. هم‌چنین به این دلیل که در گام دوم (یعنی در حلّ مسأله‌ی فوق) ماتریس $\mathbf{X}$ به طور کامل به‌روز نمی‌شود (چون ساپورت آن ثابت است)، مقدار به‌روز شده‌ی آن را با بالانویسِ $(k+\frac{1}{2})$ نشان داده‌ایم.

چون جواب فُرم بسته‌ای برای مسأله‌ی (۶-۲) برحسب دو متغیّر موجود در آن وجود ندارد (یا لااقل تا جائیکه نگارنده اطلاع دارد)، این مسأله را با استفاده از روش «بهینه‌سازی نوبتی» حل می‌کنیم. مینیمم کردن مسأله‌ی (۶-۲) برحسب $\mathbf{D}$ و با $\mathbf{X}$ ثابت منجر به جوابِ فُرم بسته‌ی الگوریتم MOD می‌شود؛ یعنی[۱] $\mathbf{D}=\mathbf{Y}\mathbf{X}^{\dagger}$. مرحله‌ی مینیمم‌سازی برحسب $\mathbf{X}$ و با استفاده از $\mathbf{D}$ فعلی امّا نیاز به دقت بیش‌تری دارد. برای این منظور، توجه کنید که با توجه به رابطه‌ی $\|\mathbf{A}\|_F^2=\sum_i\|\mathbf{a}_i\|_2^2$، مسأله‌ی (۶-۲) برحسب ستون‌های $\mathbf{X}$ قابل تفکیک است. به عبارت دیگر، برای به‌روز کردن (مقادیر غیرصفرِ) ماتریس $\mathbf{X}$ باید ستون‌های آن را جداگانه به‌روز کنیم. برای ستونِ نوعیِ $\mathbf{x}$ باید مسأله‌ی زیر را حل کنیم:

$$\min_{\mathbf{x}}\|\mathbf{y}-\mathbf{D}\mathbf{x}\|_2^2 \quad \text{subject to} \quad \text{supp}(\mathbf{x})=\text{supp}(\mathbf{x}^{(k)}). \tag{۶-۳}$$

این مسأله امّا باز هم ساده‌تر می‌شود. اگر مجموعه‌ی اندیس‌های متناظر با درایه‌های غیر صفر $\mathbf{x}^{(k)}$ را با ω نشان دهیم، مسأله‌ی فوق به مسأله‌ی زیر تبدیل می‌شود:

$$\min_{\mathbf{x}_r}\|\mathbf{y}-\mathbf{D}_r\mathbf{x}_r\|_2^2, \tag{۶-۴}$$

که $\mathbf{x}_r$ بُرداری متناظر با درایه‌های غیر صفر $\mathbf{x}^{(k)}$ و $\mathbf{D}_r$ نیز شامل ستون‌های با اندیس‌های ω از $\mathbf{D}$ است. جواب مسأله‌ی فوق بصورت زیر بدست می‌آید:

$$\mathbf{x}_r^*=(\mathbf{D}_r^T\mathbf{D}_r)^{-1}\mathbf{D}_r\mathbf{y}=\mathbf{D}_r^{\dagger}\mathbf{y}. \tag{۶-۵}$$

با توجه به اینکه ممکن است ستون‌های $\mathbf{D}_r$ خیلی شبیه هم باشند، در نتیجه، این امر منجر به بدحالت شدن ماتریس $\mathbf{D}_r^T\mathbf{D}_r$ می‌شود. به همین دلیل، باید جمله‌ی $\lambda\mathbf{I}$ را که λ عدد مثبتِ کوچکی است به این ماتریس اضافه کنیم. تعداد کمی تناوب بین به‌روز کردن دیکشنری و به‌روز کردن درایه‌های غیر صفر ماتریس ضرایب کافی است. ما در شبیه‌سازی‌ها این تعداد را ۳ گرفته‌ایم. شرط توقف الگوریتم یا می‌تواند حداکثر تعداد تناوبِ بین دو گام آموزش دیکشنری باشد و یا معیار $\|\mathbf{D}^{(k+1)}-\mathbf{D}^{(k)}\|_F\leq\epsilon$. شبه کُد این الگوریتم در شکل ۶-۱ آورده شده است.

[۱] برای سادگی، بالانویس‌ها را حذف کرده‌ایم.

- **هدف:** آموزش دیکشنری برای داده‌های $\mathbf{Y}$
- **مقداردهی اولیه:** $\mathbf{D} = \mathbf{D}^{(0)}$
- **انجام دو گام کلّیِ آموزش دیکشنری:** قرار بده $k = 0$ و گام‌های زیر را تا رسیدن به شرط توقف تکرار کن:
  ۱. **گام تقریب تُنُک:** $\mathbf{X}^{(k+1)} = \mathcal{SA}(\mathbf{Y}, \mathbf{D}^{(k)})$
  ۲. **گام به‌روز کردن دیکشنری:** قرار بده $\mathbf{X} = \mathbf{X}^{(k+1)}$ و دو گام زیر را چندبار (۳ بار کافی است) تکرار کن:
     ۱- $\mathbf{D} = \mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}$
     ۲- $\forall i: \ \mathbf{X}(\omega_i, i) = (\mathbf{D}_r^T\mathbf{D}_r + \lambda\mathbf{I})^{-1}\mathbf{D}_r\mathbf{y}_i$
  ۳. **چک کردن شرط توقف:** اگر شرط توقف برآورده نشده است قرار بده $k = k + 1$، $\mathbf{D}^{(k)} = \mathbf{D}$ و برگرد به گام تقریب تُنُک.
- **خروجی:** $\mathbf{D}$

شکل ۶-۱: الگوریتم DL1.

نسخه‌ی سریع الگوریتم DL1

الگوریتم DL1 برای مسائل با ابعاد خیلی بالا مناسب نیست. دلیل این امر نیاز به محاسبه‌ی معکوس ماتریس در گام ۲ شکل ۶-۱ است. دقت کنید که در این گام علاوه بر اینکه باید برای به‌روز کردن دیکشنری چند بار (در این‌جا ۳ بار) معکوس یک ماتریس محاسبه شود، برای به‌روز کردن درایه‌های غیر صفر ماتریس ضرایب هم باید هر بار به تعداد کلّ داده‌های آموزشی، یعنی L، معکوس ماتریس محاسبه شود. از آنجائیکه در برخی کاربردها، بخصوص پردازش تصاویر (به عنوان مثال نویززدائی از تصاویر)، تعداد داده‌های آموزشی از مرتبه‌ی چند ده‌هزار است، بنابراین عملاً گام ۲ این الگوریتم حجم محاسبات و زمان زیادی صرف می‌کند.

برای به‌روز کردن دیکشنری به جای محاسبه‌ی معکوس ماتریس، می‌توانیم از الگوریتم MM، که در فصل ۳ معرفی شد، استفاده کنیم. برای به‌روز کردن درایه‌های غیرصفرِ هر ستون از ماتریس $\mathbf{X}$ نیز می‌توانیم از روش‌های تکراری، مانند الگوریتم گرادیان-مزدوج[۱] (CG)، استفاده کنیم. راه بهتر امّا به این ترتیب است که به جای به‌روز کردن درایه‌های غیرصفر ستون‌های $\mathbf{X}$، درایه‌های غیرصفر «سطرهای» این ماتریس را به‌روز کنیم. این کار را برای هر سطر جداگانه (با ثابت نگه داشتن بقیه‌ی سطرها) انجام می‌دهیم. اگر مجموعه‌ی اندیس‌های درایه‌های غیرصفر

[۱] Conjugate Gradient

- **هدف:** آموزش دیکشنری برای داده‌های $\mathbf{Y}$
- **مقداردهی اولیه:** $\mathbf{D} = \mathbf{D}^{(0)}$
- **انجام دو گام کلّیِ آموزش دیکشنری:** قرار بده $k = 0$ و گام‌های زیر را تا رسیدن به شرط توقف تکرار کن:
  ۱. **گام تقریب تُنُک:** $\mathbf{X}^{(k+1)} = \mathcal{SA}(\mathbf{Y}, \mathbf{D}^{(k)})$
  ۲. **گام به‌روز کردن دیکشنری:** قرار بده $\mathbf{X} = \mathbf{X}^{(k+1)}$ و دو گام زیر را چندبار (۳ بار کافی است) تکرار کن:
     ۱- $\mathbf{D} = \mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}$ (یا در ابعاد بالا با استفاده از الگوریتم MM)
     ۲- برای $i = 1, \ldots, m$ مراحل زیر را انجام بده:
        - $\mathbf{E}_i = \mathbf{Y} - \sum_{j \neq i} \mathbf{d}_j \mathbf{x}_T^j$ سپس $\mathbf{E}_i^r = \mathbf{E}_i^{\Omega_i}$
        - $\mathbf{X}(i, \Omega_i) = \mathbf{d}_i^T \mathbf{E}_i^r$
  ۳. **چک کردن شرط توقف:** اگر شرط توقف برآورده نشده است قرار بده $k = k + 1$، $\mathbf{D}^{(k)} = \mathbf{D}$ و برگرد به گام تقریب تُنُک.
- **خروجی:** $\mathbf{D}$

شکل ۶-۲: الگوریتم DL2، نسخه‌ی سریعِ الگوریتم DL1.

سطر $\mathbf{x}_T^i$ را با Ω_i نشان دهیم، برای به‌روز کردن این درایه‌ها باید مسأله‌ی زیر را حل کنیم:

$$\min_{\mathbf{x}_T^r} \|\mathbf{E}_i^r - \mathbf{d}_i \mathbf{x}_T^r\|_F^2, \tag{۶-۶}$$

که در آن $\mathbf{x}_T^r$ بُرداری سطری به طول $|\Omega_i|$ بوده و $\mathbf{E}_i^r$ شامل ستون‌هائی از ماتریس $\mathbf{E}_i = \mathbf{Y} - \sum_{j \neq i} \mathbf{d}_j \mathbf{x}_T^j$ متناظر با Ω_i است. جواب مسأله‌ی فوق براحتی و با توجه به این نکته که نُرم همه‌ی اتم‌ها واحد است، بصورت زیر بدست می‌آید:

$$\mathbf{X}(i, \Omega_i) = \mathbf{d}_i^T \mathbf{E}_i^r. \tag{۶-۷}$$

به این ترتیب، به جای محاسبه‌ی L معکوس ماتریس، $m \ll L$ ضرب بُردار-ماتریس انجام می‌دهیم. توجه کنید که برای محاسبه‌ی ماتریس خطای متناظر با هر سطر، از مقادیر به‌روز شده‌ی سطرهای قبلی استفاده می‌شود. این الگوریتم در شکل ۶-۲ خلاصه شده است.

۶-۳ الگوریتم به‌روز کردن موازی اتم‌ها

در این بخش یک الگوریتم کارآ، که می‌تواند جایگزینی برای K-SVD باشد، معرفی می‌کنیم. برای سادگی، سطر متناظر با هر اتم در ماتریس ضرایب را نمایه‌ی[۱] آن اتم می‌نامیم؛ چرا که این سطر نشان می‌دهد چه داده‌هائی در نمایش تُنُک خود از آن اتم استفاده کرده‌اند. مشکل اصلی الگوریتم K-SVD حجم محاسبات زیاد آن (خصوصاً در ابعاد بالا) به دلیل محاسبه‌ی SVD ماتریس است. در [۵۲] برای به‌روز کردن هر اتم به همراه درایه‌های غیرصفر نمایه‌ی آن از یک تناوبِ بهینه‌سازی نوبتی استفاده شده است. به عبارت دقیق‌تر، به عنوان تقریبی از جواب مسأله‌ی زیر:

$$\min_{\mathbf{d},\mathbf{x}_r} \|\mathbf{E}_r - \mathbf{d}\mathbf{x}_r\|_F^2 \quad \text{subject to} \quad \|\mathbf{d}\|_2 = 1, \tag{۶-۸}$$

ابتدا اتم را بصورت $\mathbf{d} = \text{normalize}(\mathbf{E}_r\mathbf{x}_r^T)$ به‌روز کرده و سپس مقادیر به‌روز شده‌ی درایه‌های غیرصفر نمایه‌ی آن را بصورت $\mathbf{x}_r = \mathbf{d}^T\mathbf{E}_r$ محاسبه می‌کنیم. به این ترتیب، یک پیاده‌سازی تقریبی و خیلی سریع از K-SVD بدست می‌آید.

اگرچه با انجام تعداد تناوب بیش‌تر برای حل مسأله‌ی (۶-۸) جواب‌های دقیق‌تری بدست می‌آوریم، امّا باز هم عملکرد متوسط این روش بهتر از استفاده از SVD نخواهد بود. الگوریتم پیشنهادی ما برای حلّ (۶-۸) همان روش بهینه‌سازی تناوبی است؛ امّا با شیوه‌ای متفاوت. برای توضیح این روش، فرض کنید برای به‌روز کردن هر اتم به همراه درایه‌های غیرصفر نمایه‌ی آن، تعداد K تناوب انجام دهیم. بعد از انجام این K تناوب، ماتریس $\mathbf{A}_i = \mathbf{d}_i\mathbf{x}_T^i$ (البته تنها ستون‌های متناظر با درایه‌های غیرصفرِ سطر $\mathbf{x}_T^i$؛ بقیه‌ی ستون‌های آن صفر باقی می‌مانند) به عنوان تقریبی رُتبه-۱ از $\mathbf{E}_i^r$ به‌روز می‌شود. پیشنهاد ما در این جا به این ترتیب است که به جای آنکه هر ماتریسِ $\mathbf{A}_i$ را جداگانه با انجام K تناوب به‌روز کنیم، این کار را برای همه‌ی این ماتریس‌ها بصورت موازی با هم انجام دهیم. به عبارت بهتر، در هر یک از K تناوب، همه‌ی ماتریس‌های $\mathbf{A}_i$ را موازی یکدیگر به‌روز می‌کنیم. توجیه این کار به این ترتیب است که با توجه به رابطه‌ی زیر:

$$\mathbf{E} = \mathbf{Y} - \mathbf{D}\mathbf{X} = \mathbf{Y} - (\mathbf{A}_1 + \mathbf{A}_2 + \ldots + \mathbf{A}_m), \tag{۶-۹}$$

در روش معمولی، برای به‌روز کردن به عنوان مثال $\mathbf{A}_3$، دو ماتریس $\mathbf{A}_1$ و $\mathbf{A}_2$ قبلاً به طور کامل (یعنی با انجام K تناوب) به‌روز شده‌اند؛ در حالیکه ماتریس‌های $\mathbf{A}_4, \ldots, \mathbf{A}_m$ هنوز مقادیر مربوط به گام ۱ آموزش دیکشنری را

[۱] Profile

دارند و بعبارت دیگر اصلاً به‌روز نشده‌اند. در روش پیشنهادی، ما این K تناوب را بصورت موازی برای همه‌ی این ماتریس‌ها انجام می‌دهیم. در نتیجه، هنگام به‌روز کردن هر ماتریس، بقیه‌ی ماتریس‌ها همگی تا حدّی به‌روز شده‌اند[۱]. به همین دلیل ما این الگوریتم را «آموزش دیکشنری با به‌روز کردن موازی اتم‌ها»[۲] (PAU-DL) نام نهاده‌ایم.

قبل از این که الگوریتم نهائی را بیان کنیم، توجه کنید که برای به‌روز کردن هر ماتریسِ $\mathbf{A}_i$ لازم است ماتریس $\mathbf{E}_i = \mathbf{Y} - \sum_{j \neq i} \mathbf{A}_j$ را محاسبه کنیم. برای این منظور لازم نیست هر بار این ماتریس را از نو محاسبه کنیم؛ بلکه به راحتی می‌توان نشان داد که کافی است با شروع از (۶–۹)، برای به‌روز کردن $\mathbf{A}_1$ ابتدا اثر آن را از $\mathbf{E}$ حذف کرده، در انتها برای به‌روز کردن $\mathbf{A}_2$، مقدار به‌روز شده‌ی $\mathbf{A}_1$ (که آن را با $\mathbf{A}_1^*$ نشان می‌دهیم) را به $\mathbf{E}$ اضافه کنیم[۳]. به همین ترتیب، این روال را برای به‌روز کردن بقیه‌ی ماتریس‌ها تکرار می‌کنیم. بعبارت دیگر، در گام به‌روز کردن $\mathbf{A}_i$ عملیات زیر را انجام می‌دهیم:

$$\mathbf{E}_i = \mathbf{E} + \mathbf{A}_i \;\longrightarrow\; \mathbf{E} = \mathbf{E}_i - \mathbf{A}_i^*. \qquad (۶–۱۰)$$

شکل ۶–۳ الگوریتم نهائی PAU-DL را نشان می‌دهد. در این الگوریتم انتخاب مقدار K برابر با ۳ کفایت می‌کند.

۶–۴ الگوریتم RLMC-DL[۴]

الگوریتمی که در این بخش معرفی می‌کنیم مبتنی بر رگولاریزیشن مسأله‌ی به‌روز کردن دیکشنری (یعنی حداقل کردن خطای کلّی تقریب تُنُک) بوده که مشابه MOD منجر به یک جواب فُرم بسته می‌شود. برای این منظور، جمله‌ای به تابع هدف مسأله‌ی مذکور اضافه می‌کنیم که همزمان هم از بروز جواب‌های نامطلوب به دلیل بدحالت بودن ماتریسِ $\mathbf{X}\mathbf{X}^T$ جلوگیری می‌کند و هم باعث کاهش همبستگی متقابل دیکشنری نهائی می‌شود.

ایده‌ی رگولاریزه کردن مسأله‌ی به‌روز کردن دیکشنری در تعدادی از مقالات از جمله [۶۱] و [۴۰] استفاده شده است. همانطور که در زیربخشِ ۳–۵–۱ اشاره شد، این کار در واقع معادل با فرض یک اطّلاعات پیشین در مورد دیکشنری است. ساده‌ترین رگولاریزیشن شاید همانی است که در زیربخشِ ۳–۵–۲ بیان شد که البته در [۶۱]

[۱] البته این گفته برای تناوب‌های ۲ به بعد درست است.

[۲] Parallel Atom-Updating Dictionary Learning

[۳] به این نکته در [۱] و [۵۲] و حتّی در پیاده‌سازی K-SVD توجه نشده است.

[۴] Regularized Low Mutual Coherence DL

- **هدف:** آموزش دیکشنری برای داده‌های $\mathbf{Y}$
- **مقداردهی اولیه:** $\mathbf{D} = \mathbf{D}^{(0)}$
- **انجام دو گام کلّیِ آموزش دیکشنری:** قرار بده $k = 0$ و گام‌های زیر را تا رسیدن به شرط توقف تکرار کن:
  ۱. **گام تقریب تُنُک:** $\mathbf{X}^{(k+1)} = \mathcal{SA}(\mathbf{Y}, \mathbf{D}^{(k)})$
  ۲. **گام به‌روز کردن دیکشنری:** قرار بده $\mathbf{X} = \mathbf{X}^{(k+1)}$، $\mathbf{D} = \mathbf{D}^{(k)}$ و $\mathbf{E} = \mathbf{Y} - \mathbf{DX}$ و گام زیر را K بار تکرار کن:
     - مراحل زیر را برای $i = 1, \ldots, m$ انجام بده:
       ۱- $\mathbf{E}_i = \mathbf{E} + \mathbf{A}_i$ سپس $\mathbf{E}_i^r = \mathbf{E}_i^{\Omega_i}$
       ۲- $\mathbf{d}_i = \text{normalize}(\mathbf{E}_i^r \mathbf{x}_r^T)$
       ۳- $\mathbf{x}_T^i(\Omega_i) = \mathbf{d}_i^T \mathbf{E}_i^r$ سپس $\mathbf{A}_i = \mathbf{d}_i \mathbf{x}_T^i$
       ۴- $\mathbf{E} = \mathbf{E}_i - \mathbf{A}_i$
  ۳. **چک کردن شرط توقف:** اگر شرط توقف برآورده نشده است قرار بده $k = k + 1$، $\mathbf{D}^{(k)} = \mathbf{D}$ و برگرد به گام تقریب تُنُک.
- **خروجی:** $\mathbf{D}$

شکل ۶-۳: الگوریتم PAU-DL.

نیز ذکر شده است. در این روش، اطّلاعات پیشینِ راجع به دیکشنری، که همان محدود بودن نُرم فروبینیوس آن است، بصورت یک جمله به تابع هدف مسأله اضافه شده است. قید دیگری که به عنوان اطّلاعات پیشینِ دیکشنری در [۶۱] استفاده شده است، محدود بودن نُرم اتم‌های دیکشنری است که منجر به مسأله‌ی زیر می‌شود:

$$\min_{\mathbf{D}} \|\mathbf{Y} - \mathbf{DX}\|_F^2 + \sum_i \lambda_i \|\mathbf{d}_i\|_2^2, \tag{۶-۱۱}$$

که در آن λ_i پارامتر رگولاریزیشنِ مربوط به اتم iاُم است. با تعریف $\mathbf{\Lambda} = \text{diag}(\lambda_i)$، جواب این مسأله بصورت فُرم بسته‌ی زیر بدست می‌آید:

$$\mathbf{D}^* = \mathbf{YX}^T(\mathbf{XX}^T + \mathbf{\Lambda})^{-1}. \tag{۶-۱۲}$$

مسأله‌ای که ما در نظر می‌گیریم بصورت زیر است:

$$\min_{\mathbf{D}} \|\mathbf{Y} - \mathbf{DX}\|_F^2 + \frac{\lambda}{2}\|\mathbf{D}^T\mathbf{D}\|_F^2. \tag{۶-۱۳}$$

ماتریس $\mathbf{G} = \mathbf{D}^T\mathbf{D}$ به «ماتریس گرامِ»[1] $\mathbf{D}$ معروف بوده که درایه‌های آن عبارت‌اند از $g_{ij} = \mathbf{d}_i^T\mathbf{d}_j$. این ماتریس در مباحث تئوری نمایش تُنُک نقشی اساسی ایفا می‌کند؛ چرا که داریم:

[1] Gramm matrix

$$\mu(\mathbf{D}) = \max_{i \neq j} |g_{ij}|. \quad (۱۴-۶)$$

همبستگی متقابل دیکشنری یا همان $\mu(\mathbf{D})$، همانطور که در فصل ۲ با بیان قضایای مربوط به یکتائی نمایش تُنُک دیدیم، در بازیابی نمایش تُنُک سیگنال‌ها نقش بسزائی دارد. هرچه مقدار این پارامتر کم‌تر باشد، محدودیتِ روی تعداد درایه‌های غیرصفر تُنُک‌ترین جواب سبک‌تر خواهد بود. به علاوه، این پارامتر نقش مهمی در موفقیت بسیاری از الگوریتم‌های بازیابی نمایش تُنُک، بخصوص الگوریتم OMP دارد [۵۵]. هرچه همبستگی متقابل دیکشنری کم‌تر باشد، به این معنی است که شباهت دو اتمِ متمایز به یکدیگر کم‌تر است. در نتیجه، الگوریتم حریصی مانند OMP در انتخاب اتمِ مناسب کم‌تر دچار اشتباه خواهد شد.

جمله‌ی $\|\mathbf{D}^T\mathbf{D}\|_F^2$ در رابطه‌ی (۶-۱۳) هم روی نُرم اتم‌ها جریمه قرار می‌دهد، مانند مسأله‌ی (۶-۱۱)، و هم روی همبستگی متقابل اتم‌ها. برای توضیح بیش‌تر، دقت کنید که مسأله‌ی (۶-۱۳) معادل مسأله‌ی زیر است:

$$\min_{\mathbf{D}} \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 + \frac{\lambda}{2} \sum_i \|\mathbf{d}_i\|_2^4 + \frac{\lambda}{2} \sum_{i \neq j} (\mathbf{d}_i^T \mathbf{d}_j)^2. \quad (۱۵-۶)$$

برای حل مسأله‌ی (۶-۱۳)، مشتق تابع هدف را برابر با صفر قرار می‌دهیم. در اینصورت بدست می‌آوریم:

$$-(\mathbf{Y} - \mathbf{D}\mathbf{X})\mathbf{X}^T + \lambda \mathbf{D}(\mathbf{D}^T\mathbf{D}) = 0. \quad (۱۶-۶)$$

معادله‌ی فوق جواب فُرم بسته‌ای برای دیکشنری ندارد و در حالت کلّی باید از الگوریتم‌های تکراری، مانند الگوریتم گرادیانِ کاهشی، برای حلّ آن استفاده کرد. ما برای حلّ معادله‌ی فوق مشابه یک دسته از روش‌های بازگشتی در بهینه سازی، موسوم به «روش‌های نقطه‌ی ثابت»[۱]، معادله‌ی زیر را برای $\mathbf{D}^{(t+1)}$ حل می‌کنیم:

$$-(\mathbf{Y} - \mathbf{D}^{(t+1)}\mathbf{X})\mathbf{X}^T + \lambda \mathbf{D}^{(t+1)}\mathbf{G}^{(t)} = 0. \quad (۱۷-۶)$$

که در آن $\mathbf{G}^{(t)}$ ماتریس گرام $\mathbf{D}^{(t)}$ است. جواب معادله‌ی فوق بصورت فُرم بسته‌ی زیر بدست می‌آید[۲]:

$$\mathbf{D}^* = \mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda \mathbf{D}^T\mathbf{D})^{-1}. \quad (۱۸-۶)$$

دقت کنید که به این ترتیب ما تنها «یک» تکرار از معادله‌ی (۶-۱۷) را انجام داده‌ایم. $\mathbf{D}$ و $\mathbf{X}$ موجود در سمت راست معادله‌ی فوق به ترتیب دیکشنریِ فعلی و ماتریس ضرایب گام نخست بوده و سمت چپ این معادله دیکشنری به‌روز شده در گام دوم است. به تفاوت این معادله با معادله‌ی (۶-۱۲) دقت کنید.

دیدیم که معادله‌ی (۶-۱۲) جواب مسأله‌ی (۶-۱۱) است و همینطور در زیربخش ۳-۵-۲، معادله‌ی (۳-۱۲) جواب مسأله‌ی (۳-۱۳) است. امّا معادله‌ی (۶-۱۸) جواب چه مسأله‌ای است؟ برای پاسخ به این سؤال، ابتدا توجه

[۱] Fixed-point method

[۲] برای سادگی، اندیس‌ها را حذف کرده‌ایم.

کنید که داریم $\|\mathbf{A}\|_F^2 = \mathrm{Tr}(\mathbf{A}\mathbf{A}^T)$ و نیز $\mathrm{Tr}(\mathbf{A}^T) = \mathrm{Tr}(\mathbf{A})$. یکی از خواص ردّ ماتریس، که در ادامه از آن استفاده خواهیم کرد، بصورت زیر است:

$$\mathrm{Tr}(\mathbf{ABC}) = \mathrm{Tr}(\mathbf{BCA}) = \mathrm{Tr}(\mathbf{CAB}). \tag{۱۹–۶}$$

با این توضیح داریم:

$$\|\mathbf{D}^T\mathbf{D}\|_F = \mathrm{Tr}(\mathbf{D}^T\mathbf{D}\mathbf{D}^T\mathbf{D}). \tag{۲۰–۶}$$

به این ترتیب، معادله‌ی (۶–۱۷) در واقع مسأله‌ی زیر را حل می‌کند:

$$\mathbf{D}^{(t+1)} = \arg\min_{\mathbf{D}} \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 + \frac{\lambda}{2}\|\mathbf{D}(\mathbf{D}^{(t)})^T\|_F^2. \tag{۲۱–۶}$$

حال با توجه به رابطه‌ی زیر

$$\mathrm{Tr}(\mathbf{D}^T\mathbf{D}\mathbf{D}^T\mathbf{D}) = \mathrm{Tr}(\sum_{i=1}^{n}\mathbf{p}_i\mathbf{p}_i^T\mathbf{D}^T\mathbf{D}) = \sum_{i=1}^{n}\mathrm{Tr}(\mathbf{p}_i\mathbf{p}_i^T\mathbf{D}^T\mathbf{D}) = \sum_{i=1}^{n}\mathbf{p}_i^T\mathbf{D}^T\mathbf{D}\mathbf{p}_i, \tag{۲۲–۶}$$

که در آن $\mathbf{p}_i$ ستون i اُم ماتریس $\mathbf{D}^T$ است، بدست می‌آوریم:

$$\mathbf{D}^* = \arg\min_{\mathbf{D}} \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 + \frac{\lambda}{2}\sum_{i=1}^{n}\mathbf{p}_i^T\mathbf{D}^T\mathbf{D}\mathbf{p}_i, \tag{۲۳–۶}$$

که در آن همانطور که پیش‌تر گفته شد $\mathbf{D}^* = \mathbf{D}^{(k+1)}$، $\mathbf{p}_i$ ستون i اُم ترانهاده‌ی ماتریس $\mathbf{D}^{(k)}$ و k شماره‌ی تناوب است. شبه کد این الگوریتم در شکل ۶–۴ آمده است.

در ادامه، برای آشنائی بیشتر با روش‌های نقطه‌ی ثابت، نشان می‌دهیم که معادله‌ی بازگشتیِ (۳–۲۵) مربوط به الگوریتم MM-DL را به راحتی می‌توان با استفاده از این رهیافت بدست آورد. برای این منظور، مشابه الگوریتم MOD برای به‌روز کردن دیکشنری، از تابع خطای کلّی نسبت به $\mathbf{D}$ مشتق می‌گیریم. در این صورت بدست می‌آوریم $\mathbf{D}\mathbf{X}\mathbf{X}^T = \mathbf{Y}\mathbf{X}^T$. حال، جمله‌ی $c\mathbf{D}$ را به دو طرف این معادله اضافه می‌کنیم. تا اینجا داریم:

$$\mathbf{D}\mathbf{X}\mathbf{X}^T + c\mathbf{D} = \mathbf{Y}\mathbf{X}^T + c\mathbf{D}. \tag{۲۴–۶}$$

با تغییر آرایش جملات معادله‌ی فوق بدست می‌آوریم:

$$\mathbf{D} = \frac{1}{c}(\mathbf{Y}\mathbf{X}^T + \mathbf{D}(c\mathbf{I} - \mathbf{X}\mathbf{X}^T)). \tag{۲۵–۶}$$

استفاده از ایده‌ی روش‌های نقطه‌ی ثابت برای حلّ تکراریِ معادله‌ی فوق دقیقاً منجر به معادله‌ی (۳–۲۵) می‌شود. واضح است که برای همگرائی این الگوریتم، ثابتِ c هر عددی نمی‌تواند باشد، بلکه باید $c > \lambda_{\max}(\mathbf{X}^T\mathbf{X})$.

- **هدف:** آموزش دیکشنری برای داده‌های $\mathbf{Y}$
- **مقداردهی اولیه:** $\mathbf{D} = \mathbf{D}^{(0)}$
- **انجام دو گام کلّیِ آموزش دیکشنری:** قرار بده $k = 0$ و گام‌های زیر را تا رسیدن به شرط توقف تکرار کن:
  ۱. **گام تقریب تُنُک:** $\mathbf{X}^{(k+1)} = \mathcal{SA}(\mathbf{Y}, \mathbf{D}^{(k)})$
  ۲. **گام به‌روز کردن دیکشنری:** قرار بده $\mathbf{X} = \mathbf{X}^{(k+1)}$، $\mathbf{D} = \mathbf{D}^{(k)}$ و سپس:
  $$\mathbf{D}^{(k+1)} = \mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{D}^T\mathbf{D})^{-1}$$
  ۳. **چک کردن شرط توقف:** اگر شرط توقف برآورده نشده است قرار بده $k = k + 1$ و برگرد به گام تقریب تُنُک.
- **خروجی:** $\mathbf{D}$

شکل ۶-۴: الگوریتم RLMC-DL.

۶-۵ الگوریتم OS-DL[1]

الگوریتم پیشنهادی این بخش از نظر شیوه‌ی کلّی حلّ مسائل آموزش دیکشنری، تفاوتی اساسی با الگوریتم‌های موجود دارد. به بیان دقیق‌تر، در این الگوریتم خبری از گام نخست، یعنی بدست آوردن نمایش تُنُک داده‌ها نیست؛ یا لااقل در این‌جا این کار به شیوه‌ای دیگر انجام می‌شود. ایده‌ی این الگوریتم به این ترتیب است که ابتدا بعنوان مقداردهی اولّیه‌ی، یک دیکشنری و یک ماتریس ضرایب انتخاب می‌کنیم. سپس به صورت تکراری، اتم‌های دیکشنری را یکی یکی به همراه نمایه‌ی آن‌ها به‌روز می‌کنیم. زمانی که این کار را برای همه‌ی اتم‌های دیکشنری انجام دادیم، دوباره همین روال را تکرار می‌کنیم؛ یعنی از دیکشنری و ماتریس ضرایب بدست آمده استفاده کرده و دوباره فرآیند به‌روز کردن نوبتی اتم‌های دیکشنری را تکرار می‌کنیم. کلّیت این کار شبیه گام دوم الگوریتم‌های K-SVD و PUA-DL است؛ امّا تفاوت اساسی در این‌جا است که بر خلاف این الگوریتم‌ها، در الگوریتم پیشنهادی «همه‌ی» درایه‌های نمایه‌ی هر اتم (و نه فقط درایه‌های غیرصفر آن) اجازه دارند تغییر کرده و در نتیجه به‌روز شوند. برای جلوگیری از پُر شدن همه‌ی درایه‌ها، یک قید تُنُک بودن روی نمایه‌ی هر اتم می‌گذاریم. چون در این الگوریتم تنها یک گام از دو گامِ آموزش دیکشنری انجام می‌شود، این الگوریتم را «آموزش یک مرحله‌ایِ دیکشنری» یا به اختصار OS-DL نامیده‌ایم. در ادامه توصیف دقیق این الگوریتم را بیان می‌کنیم.

[1] One Stage DL

مسأله‌ی به‌روز کردن اتمِ $\mathbf{d}_i$ به همراه نمایه‌ی آن، یعنی $\mathbf{x}_T^i$، در الگوریتم OS-DL بصورت زیر است:

$$\{\mathbf{d}_i^*, \mathbf{x}_T^{i*}\} = \arg\min_{\mathbf{d},\mathbf{x}_T} \frac{۱}{۲}\|\mathbf{E}_i - \mathbf{d}\mathbf{x}_T\|_F^۲ + \lambda\|\mathbf{x}_T\|_۱ \quad \text{subject to} \quad \|\mathbf{d}_i\|_۲ = ۱, \tag{۲۶–۶}$$

که در آن مشابه قبل $\mathbf{E}_i = \mathbf{Y} - \sum_{j\neq i}\mathbf{d}_j\mathbf{x}_T^j$. دقت کنید که مسأله‌ی فوق در حقیقت تقریب رُتبه–یک ماتریس $\mathbf{E}_i$ بوده که مقید به تُنُک بودن بُردار $\mathbf{x}_T$ است. اگر قید نُرم یکِ بُردار $\mathbf{x}_T$ را برداریم، همانطور که در الگوریتم K-SVD دیدیم، این مسأله جوابِ فُرم بسته داشته و از تجزیه مقادیر تکین ماتریس $\mathbf{E}_i$ بدست می‌آید. استفاده از قید نُرمِ یک در این مسأله کمک می‌کند که بردار $\mathbf{x}_T$ تُنُک باشد. برای حل مسأله‌ی (۶–۲۶) از بهینه‌سازی نوبتی استفاده می‌کنیم؛ یعنی ابتدا برُدار $\mathbf{d}$ را ثابت گرفته، مسأله را نسبت به بُردار $\mathbf{x}_T$ حل می‌کنیم و بالعکس. وقتی بُردار $\mathbf{d}$ ثابت است، باید مسأله‌ی زیر را برای $\mathbf{x}_T$ حل کنیم:

$$\mathbf{x}_T^{i*} = \arg\min_{\mathbf{x}_T} \frac{۱}{۲}\|\mathbf{E}_i - \mathbf{d}\mathbf{x}_T\|_F^۲ + \lambda\|\mathbf{x}_T\|_۱. \tag{۲۷–۶}$$

در ادامه نشان می‌دهیم که جواب مسأله‌ی فوق را به آسانی می‌توان با استفاده از تابع آستانه گذاری نَرم بدست آورد. برای این منظور، دقت کنید که این مسأله برای درایه‌های بُردار $\mathbf{x}_T$ قابل تفکیک است؛ چرا که داریم:

$$\frac{۱}{۲}\|\mathbf{E}_i - \mathbf{d}\mathbf{x}_T\|_F^۲ + \lambda\|\mathbf{x}_T\|_۱ = \sum_{l=۱}^{L}\left\{\frac{۱}{۲}\|\mathbf{e}_i^l - \mathbf{d}\mathbf{x}_T(l)\|_۲^۲ + \lambda|\mathbf{x}_T(l)|\right\}, \tag{۲۸–۶}$$

که در آن، $\mathbf{e}_i^l$ ستون l اُم ماتریس $\mathbf{E}_i$ است. به این ترتیب، باید تعدادی مسأله با متغیر اسکالر به فُرم زیر را حل کنیم:

$$x^* = \arg\min_x \frac{۱}{۲}\|\mathbf{e} - \mathbf{d}x\|_۲^۲ + \lambda|x|. \tag{۲۹–۶}$$

برای حلّ معادله‌ی فوق باید گرادیان تابع هدف را برابر با صفر قرار دهیم. چون تابع قدرمطلق مشتق‌پذیر نیست، باید از زیرگرادیان آن، که همان تابع علامت[۱] است، استفاده کنیم. در اینصورت بدست می‌آوریم:

$$-\mathbf{d}^T(\mathbf{e} - x^*\mathbf{d}) + \lambda\text{sgn}(x^*) = ۰. \tag{۳۰–۶}$$

جواب نهائی معادله‌ی فوق با استفاده از تابع آستانه‌گذاری نَرم بصورت زیر بدست می‌آید:

$$x^* = \text{sgn}(\mathbf{e}^T\mathbf{d}) \times \max(۰, |\mathbf{e}^T\mathbf{d}| - \lambda) = \mathcal{S}_\lambda(\mathbf{e}^T\mathbf{d}). \tag{۳۱–۶}$$

در نهایت، جواب مسأله‌ی (۶–۲۷) به صورت فُرمِ بسته‌ی زیر بدست می‌آید:

$$\mathbf{x}_T^{i*} = \text{sgn}(\mathbf{E}_i^T\mathbf{d}). \times \max(۰, |\mathbf{E}_i^T\mathbf{d}| - \lambda\mathbf{o}) = \mathcal{S}_\lambda(\mathbf{E}_i^T\mathbf{d}), \tag{۳۲–۶}$$

که در آن منظور از $\times$. ضرب نقطه به نقطه بوده و $\mathbf{o}$ بُرداری است که همه‌ی درایه‌های آن برابر با یک است. عملگرهای موجود در رابطه‌ی فوق به صورت درایه به درایه عمل می‌کنند. برای به روز کردن بُردار $\mathbf{d}$ ، زمانی که

[۱] Sign function

- **هدف:** آموزش دیکشنری برای داده‌های $\mathbf{Y}$
- **مقداردهی اولیه:** $\mathbf{D} = \mathbf{D}^{(0)}$ و $\mathbf{X} = \mathbf{X}^{(0)}$
- **انجام گام دوّم آموزش دیکشنری:** قرار بده $k = 0$ و گام‌های زیر را تا رسیدن به شرط توقف تکرار کن:
  - **به‌روز کردن موازی اتم‌ها و نمایه‌ی آن‌ها:** قرار بده $\mathbf{E} = \mathbf{Y} - \mathbf{DX}$ و گام زیر را K بار تکرار کن:
    - مراحل زیر را برای $i = 1, \ldots, m$ انجام بده:
      ۱- $\mathbf{E}_i = \mathbf{E} + \mathbf{A}_i$
      ۲- $\mathbf{x}_T^i = \mathcal{S}_\lambda(\mathbf{E}_i^T \mathbf{d}_i)$
      ۳- $\mathbf{d}_i = \text{normalize}(\mathbf{E}_i(\mathbf{x}_T^i)^T)$ سپس $\mathbf{A}_i = \mathbf{d}_i \mathbf{x}_T^i$
      ۴- $\mathbf{E} = \mathbf{E}_i - \mathbf{A}_i$
  - **چک کردن شرط توقف:** اگر شرط توقف برآورده نشده است قرار بده $k = k + 1$ و برگرد به گام به‌روز کردن اتم‌ها.
- **خروجی:** $\mathbf{D}$

شکل ۶-۵: الگوریتم OSP-DL.

بُردار $\mathbf{x}_T$ ثابت است، همانطور که از قبل می‌دانیم جواب زیر بدست می‌آید:

$$\mathbf{d}_i^* = \text{normalize}(\mathbf{E}_i(\mathbf{x}_T^{i*})^T). \quad (۶-۳۳)$$

انجام ۳ تناوب بین به‌روز کردن هر اتم و نمایه‌ی آن کافی است. بعلاوه، برای افزایش سرعت همگرائی الگوریتم و بهبود کیفیتِ جواب نهائی، مشابه الگوریتم PAU-DL به‌روز کردن اتم‌ها را موازی با هم انجام می‌دهیم. در این حالت الگوریتم را OSP-DL می‌نامیم. شبه کُد این الگوریتم در شکل ۶-۵ آمده است.

سؤالی که در اینجا پیش می‌آید این است که با توجه به اینکه در الگوریتم OS-DL گام نخستِ آموزش دیکشنری انجام نمی‌شود و همه‌ی الگوریتم‌های موجود برای آموزش دیکشنری این گام را انجام می‌دهند، چرا باید انتظار داشت این الگوریتم کار کند؟ در پاسخ باید گفت که همانطور که در ابتدای این بخش اشاره شد، در حقیقت گام نخست در این الگوریتم هم انجام می‌شود امّا به شیوه‌ای متفاوت. برای توضیح بیش‌تر، دقت کنید که بعد از اینکه یک دور نمایه‌ی همه‌ی اتم‌ها به‌روز شود، همزمان ستون‌های ماتریس $\mathbf{X}$ هم به‌روز شده‌اند!

به شباهت مسأله‌ی (۶-۲۷) با کُدینگ تُنُک سیگنال $\mathbf{d}$ در دیکشنری $\mathbf{E}_i$ دقت کنید. در معادله‌ی (۶-۳۲) برای بدست آوردن $\mathbf{x}_T^{i*}$، ابتدا مقادیر همبستگی اتمِ $\mathbf{d}_i$ با ستون‌های ماتریسِ $\mathbf{E}_i$ محاسبه شده و در نهایت روی این مقادیر آستانه‌گذاری انجام می‌شود. معادله‌ی (۶-۳۳) نیز بوضوح، بیانِ اتم $\mathbf{d}_i$ را برحسب یک ترکیب خطی از

ستون‌های $\mathbf{E}_i$ نشان می‌دهد. در نتیجه طیّ این فرآیند، هر «اتم» سیگنال‌های خوشه‌ی خود را انتخاب می‌کند. از طرفی، در گام نخستِ آموزش دیکشنری، هر «سیگنال» اتم‌های مناسب برای نمایش تُنُک خود را انتخاب می‌کند. با این توضیحات، ایده‌ی ترکیب این دو گام به ذهن می‌رسد که این خود منجر به معرفی الگوریتم بخش بعدی می‌شود.

۶-۶ الگوریتم DL3

اگرچه همانطور که در بخش قبل گفتیم، گام نخستِ آموزش دیکشنری در الگوریتم OS-DL هم به نوعی انجام می‌شود، امّا اگر صریحاً این گام را هم انجام دهیم، انتظار داریم سرعت همگرائی الگوریتم افزایش یابد. در این وضعیت، الگوریتم حاصل را DL3 می‌نامیم. این الگوریتم مشابه الگوریتم MM-DL از فصل ۳ مسأله‌ی زیر را برای آموزش دیکشنری حل می‌کند:

$$\min_{\mathbf{D}\in\mathcal{D},\mathbf{X}} \|\mathbf{Y}-\mathbf{D}\mathbf{X}\|_F^2 + \lambda\|\mathbf{X}\|_1. \qquad (۳۴-۶)$$

گام نخست آموزش دیکشنری را با استفاده از الگوریتم‌های IST انجام می‌دهیم. در گام دوم، اتم‌ها را یکی یکی به همراه نمایه‌شان به‌روز می‌کنیم. با توجه به (۶-۳۴)، مسأله‌ی به‌روز کردن $\mathbf{d}_i$ به همراه $\mathbf{x}_T^i$ با ثابت گرفتن بقیه‌ی اتم‌ها و نمایه‌ی آن‌ها دقیقاً منجر به مسأله‌ی (۶-۲۶) می‌شود. به تفاوت گام دوم این الگوریتم با گام دوم الگوریتم K-SVD دقت کنید. در K-SVD برای به‌روز کردن نمایه‌ی هر اتم، تنها درایه‌های غیرصفر آن از گام نخست به‌روز می‌شوند؛ حال آنکه در الگوریتم DL3 همانطور که در بخش قبل هم اشاره شد، «کلّ» نمایه‌ی هر اتم اجازه‌ی به‌روز شدن دارد. این کار البته توجیه هم دارد. دقت کنید که طی فرآیند به‌روز کردن اتم‌ها، به احتمال زیاد ساپورت نمایه‌ی آن‌ها نیز تغییر خواهد کرد، در حالیکه در K-SVD ساپورت هر اتم به اجبار ثابت نگه داشته می‌شود. به عنوان مثال، فرض کنید ساپورت هیچکدام از اتم‌ها بعد از انجام گام دوم تغییر نکند. در نتیجه، بعد از انجام گام نخست با استفاده از دیکشنری به‌روز شده هم انتظار خواهیم داشت که ساپورت اتم‌ها تغییر نکند. بنابراین الگوریتم در واقع در حلقه‌ی یک مینیمم محلّی گرفتار شده و از آن خارج نخواهد شد. در نتیجه، فرض ثابت بودن ساپورت منتفی است.

۶-۷ الگوریتم SSF[1]

با الهام از رهیافت نمایش تُنُک و الگوریتم‌های آموزش دیکشنری، در این قسمت الگوریتمی معرفی می‌کنیم که مبتنی بر برازش متوالی چندین زیرفضا به داده‌های آموزشی است و به همین دلیل آن را «برازش متوالی زیرفضا» (SSF) نام نهاده‌ایم. این الگوریتم جزء الگوریتم‌های مبتنی بر خوشه‌بندی زیرفضا است و همانطور که در فصل ۳ بحث کردیم، کارآئی آن در مواردی است که ساختار داده‌ها واقعاً متشکل از تعدادی زیرفضا باشد. در الگوریتم SSF فرض بر این است که ابعاد زیرفضاها معلوم بوده ولی لزومی به معلوم بودن تعداد آن‌ها نیست. بدون از دست دادن کلّیت مسأله، فرض می‌کنیم که ابعاد همه‌ی زیرفضاها یکسان و برابر با s است. در ادامه‌ی بحث، $\mathbf{D}_s \in \mathbb{R}^{n\times s}$ پایه‌ی یک زیرفضا با بُعد s است و ماتریس $\mathbf{X}_s \in \mathbb{R}^{s\times L}$ نیز ضرایب داده‌های موجود در این زیرفضا را نشان می‌دهد. نُرم ستون‌های ماتریس $\mathbf{D}_s$ واحد است.

ایده‌ی الگوریتم SSF به این ترتیب است که ابتدا زیرفضای $\mathbf{D}_s$ را پیدا کرده یا به عبارتی به داده‌ها برازش می‌کنیم. نخستین راه‌حلی که به ذهن می‌رسد حلّ مسأله‌ی زیر است:

$$\min_{\mathbf{D}_s,\mathbf{X}_s} \|\mathbf{Y}-\mathbf{D}_s\mathbf{X}_s\|_F^2 = \sum_i \|\mathbf{y}_i-\mathbf{D}_s\mathbf{x}_i\|_2^2. \qquad (۳۵-۶)$$

مسأله‌ی فوق امّا زیرفضای $\mathbf{D}_s$ را به طور متوسط به داده‌ها برازش می‌کند. بعبارت دیگر، پایه‌ی $\mathbf{D}_s$ به نحوی در ساختار هندسی داده‌ها قرار داده می‌شود که خطای نمایش همه‌ی داده‌ها به طور متوسط کم باشد. این کار بسیار شبیه اِعمال PCA روی داده‌ها است. هدف ما امّا برازش چندین زیرفضا به گونه‌ای است که تا حد امکان همه‌ی ساختار هندسی داده‌ها (یعنی نمایش آن‌ها بصورت نقاطی در فضا) را بخصوص زمانی که داده‌ها واقعاً در چندین زیرفضا قرار دارند، پوشش دهند. برای این منظور، هر زیرفضا را باید طوری برازش کنیم که خطای نمایش تعدادی از داده‌ها (همان‌هائی که با هم در یک زیرفضا قرار دارند) بسیار کم بوده و بقیه احتمالاً[2] خطای زیادی داشته باشند. برای رسیدن به این هدف، مسأله‌ی زیر را به جای (۶-۳۵) حل می‌کنیم:

$$\min_{\mathbf{D}_s,\mathbf{X}_s} \|\mathbf{e}\|_0, \qquad (۳۶-۶)$$

که در آن $\mathbf{e}\in\mathbb{R}^L$ و درایه‌ی i اُم آن برابر است با $e_i = \|\mathbf{y}_i-\mathbf{D}_s\mathbf{x}_i\|_2$. به این ترتیب، زیرفضای $\mathbf{D}_s$ برای تعداد کمی از داده‌ها نمایشی با خطای کم خواهد داشت. برای حلّ مسأله‌ی (۶-۳۶) از ایده‌ای مشابه الگوریتم‌های IRLS

[1] Sequentially Subspace Fitting

[2] ممکن است داده‌ای در نزدیکی فصل مشترک دو زیرفضا بوده و بنابراین خطای نمایش آن در هردو زیرفضا کم باشد.

استفاده می‌کنیم. در این صورت باید تکرارهای زیر را انجام دهیم:

$$\left\{\mathbf{D}_s^{(k+1)}, \mathbf{X}_s^{(k+1)}\right\} = \arg\min_{\mathbf{D}_s, \mathbf{X}_s} \sum_i w_i^{(k)} \|\mathbf{y}_i - \mathbf{D}_s \mathbf{x}_i\|_2^2, \tag{۳۷–۶}$$

که $w_i^{(k)} = (e_i^{(k)} + \sigma)^{p-2}$ و $e_i^{(k)}$ خطای نمایش داده‌ی $\mathbf{y}_i$ در تکرار k اُم است. نقش σ همان رگولاریزیشن در الگوریتم‌های IRLS است. به تفاوت این مسأله با مسأله‌ی (۶–۳۵) دقت کنید. برای حلّ (۶–۳۷) مشابه الگوریتم‌های آموزش دیکشنری از مینیمم سازی نوبتی استفاده می‌کنیم. با انتخاب $\mathbf{D}_s = \mathbf{D}_s^{(k)}$، مسأله‌ی به‌روز کردن $\mathbf{X}_s$ برحسب ستون‌های آن قابل تفکیک بوده و به $w_i^{(k)}$ ها وابسته نیست. در اینصورت داریم:

$$\mathbf{X}_s^{(k+1)} = \mathbf{D}_s^{\dagger} \mathbf{Y}. \tag{۳۸–۶}$$

برای به‌روز کردن دیکشنری کافی است مشتق تابع هدف (۶–۳۷) را برابر با صفر قرار دهیم. بدست می‌آوریم:

$$\sum_i -2 w_i^{(k)} (\mathbf{y}_i - \mathbf{D}_s \mathbf{x}_i) \mathbf{x}_i^T = 0. \tag{۳۹–۶}$$

با تعریف $\mathbf{W}_k = \mathrm{diag}(w_i^{(k)})$ براحتی می‌توان نشان داد که جواب معادله‌ی فوق بصورت زیر بدست می‌آید:

$$\mathbf{D}_s^{(k+1)} = (\mathbf{Y}\mathbf{W}_k\mathbf{X}_s^T)(\mathbf{X}_s\mathbf{W}_k\mathbf{X}_s^T)^{-1}. \tag{۴۰–۶}$$

لازم است سپس ستون‌های پایه‌ی بدست آمده نُرمالیزه شود. با توجه به این که محاسبه‌ی معکوس ماتریس در ابعاد بالا حجم محاسبات زیادی می‌خواهد، مشابه الگوریتم DL2 می‌توانیم از روش‌های تکراری مانند CG یا MM استفاده کنیم. استفاده از روش MM برای محاسبه‌ی $\mathbf{D}_s^{(k+1)}$ منجر به انجام تکرارهای زیر می‌شود:

$$\mathbf{D}_s^{(t+1)} = \arg\min_{\mathbf{D}_s} \sum_i \left\{ w_i^{(k)} \|\mathbf{y}_i - \mathbf{D}_s\mathbf{x}_i\|_2^2 + c w_i^{(k)} \|\mathbf{D}_s - \mathbf{D}_s^{(t)}\|_F^2 - w_i^{(k)} \|\mathbf{D}_s\mathbf{x}_i - \mathbf{D}_s^{(t)}\mathbf{x}_i\|_2^2 \right\} \tag{۴۱–۶}$$

که در آن $c > \lambda_{\max}(\mathbf{X}_s^T\mathbf{X}_s)$ یک عدد ثابت است. جواب مسأله‌ی فوق بصورت زیر است:

$$\mathbf{D}_s^{(t+1)} = \frac{1}{\alpha}(\mathbf{Y}\mathbf{W}_k\mathbf{X}_s^T + \mathbf{D}_s^{(t)}(\alpha\mathbf{I} - \mathbf{X}_s\mathbf{W}_k\mathbf{X}_s^T)), \quad \alpha = c\sum_i w_i^{(k)} \tag{۴۲–۶}$$

در نهایت، بعد از پیدا کردن $\mathbf{D}_s$ بهینه، برای پیدا کردن زیرفضای بعدی، داده‌های موجود در زیرفضای پیدا شده را از $\mathbf{Y}$ حذف کرده و روال فوق را برای پیدا کردن دومین زیرفضا تکرار می‌کنیم. برای تشخیص اینکه آیا $\mathbf{y}_i$ در زیرفضای پیدا شده قرار دارد یا نه، خطای نمایش آن، یعنی e_i، را با یک آستانه δ مقایسه می‌کنیم. برای توضیح بیش‌تر، شکل ۶–۷ را ببینید. در این شکل تعدادی داده روی سه زیرفضای دو بُعدی قرار دارند. الگوریتم SSF بطور متوالی این زیرفضاها را پیدا کرده و هربار داده‌های موجود در زیرفضای پیدا شده را از مجموع داده‌های مورد پردازش حذف می‌کند.

بعد از پیدا کردن تمام زیر فضاها (یعنی وقتی تمام داده‌ها به یک زیرفضا تخصیص داده شدند) می‌توانیم

- **هدف:** برازش تعدادی زیرفضا با ابعاد معلوم به داده‌های $\mathbf{Y}$
- **برازش زیرفضا:** قرار بده $r = ۱$ و گام‌های زیر را تا تخصیص همه‌ی داده‌ها به یک زیرفضا انجام بده:
  الف- قرار بده $\mathbf{D}_s = \mathbf{D}_s^{(۰)}$ و چهار گام زیر را انجام بده:
  ۱- $\mathbf{X}_s = \mathbf{D}_s^{\dagger}\mathbf{Y}$
  ۲- $\forall i:\ e_i = \|\mathbf{y}_i - \mathbf{D}_s\mathbf{x}_i\|_۲$ و سپس $\mathbf{W} = \mathrm{diag}(w_i = (e_i + \sigma)^{p-۲})$
  ۳- $\mathbf{D}_s = (\mathbf{Y}\mathbf{W}\mathbf{X}_s^T)(\mathbf{X}_s\mathbf{W}\mathbf{X}_s^T)^{-۱}$ و سپس ستون‌های $\mathbf{D}_s$ را نرمالیزه کن
  ۴- اگر شرط توقف برآورده نشده است قرار بده $\sigma = \zeta\sigma$ و برگرد به گام ۱
  ب- $w = \{i:\ e_i > \delta\}$
  ج- $\mathbf{Y} = \mathbf{Y}(:, w)$
  د- $\mathbf{D}_s^r = \mathbf{D}_s$ و $r = r + ۱$ و برگرد به گام الف
- **خروجی:** $\mathbf{D}_s^r,\ \ r = ۱, ۲, \ldots$

شکل ۶-۶: الگوریتم SSF.

برای بدست آوردن تعدادی اتم، از یک الگوریتم هَرَس زیرفضاها مانند [۳۳] برای این منظور استفاده کنیم. الگوریتم نهائی در شکل ۶-۶ خلاصه شده است. در این شکل، $۰ < \zeta < ۱$ عددی ثابت برای کاهش مقدار σ در هر تکرار است. شرط توقف در گام ۴ این شکل می‌تواند یا تعداد تکرار مشخص و یا شرط $\|\mathbf{D}_s^{(k+۱)} - \mathbf{D}_s^{(k)}\|_F \leq \epsilon$ باشد.

روش هوشمندانه‌تر برای انتخاب آستانه در گام «ب» الگوریتم SSF، استفاده از «قانون اولین جهش»[۱] است که در [۵۹] استفاده شده است. برای این منظور، درایه‌های بُردار $\mathbf{e}$ را به ترتیب از کوچک به بزرگ مرتب می‌کنیم. اگر بُردار حاصل را با $\bar{\mathbf{e}}$ نشان دهیم، آنگاه کوچکترین j ای را پیدا می‌کنیم که برای آن

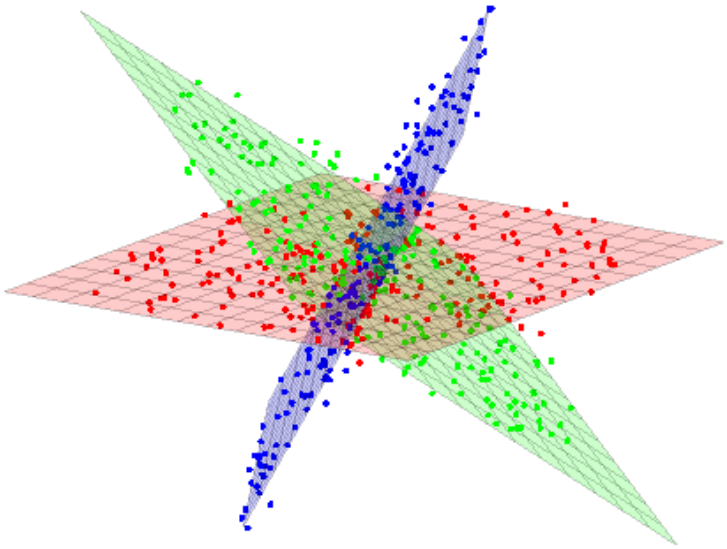
$$\bar{e}_{j+۱} - \bar{e}_j > \tau, \qquad (۴۳-۶)$$

که τ یک عدد مثبت کوچک است. اگر اندیس پیدا شده را با $j_۰$ نشان دهیم، انتخاب می‌کنیم $\delta = \bar{e}_{j_۰}$.

قبل از خاتمه‌ی این قسمت، الگوریتم K-subspace را که بی‌شباهت به الگوریتم SSF نیست و البته در شبیه‌سازی‌ها هم استفاده خواهد شد، به اختصار مرور می‌کنیم. الگوریتم K-subspace در واقع تعمیمی از الگوریتم K-means است که نخستین بار در سال ۲۰۰۳ و در [۳۶] معرفی شد. در فصل ۳ با الگوریتم K-means آشنا شدیم. این الگوریتم با پیدا کردن K بُردار (یا اتم)، در حقیقت هر داده را به یک زیرفضای «یک بُعدی» تخصیص می‌دهد. امّا عموماً داده‌ها در زیرفضاهای با بُعد بالاتر، مانند یک اَبَر صفحه[۲]، قرار دارند. الگوریتم K-subspace در جهت

[۱] First Jump Rule

[۲] Hyper-plane



شکل ۶–۷: تعدادی داده که روی سه زیرفضای دو بُعدی قرار دارند.

رفع این مشکل، به جای پیدا کردن K زیرفضای یک بُعدی، K زیرفضای s بُعدی پیدا می‌کند. رهیافت این الگوریتم همانی است که در K-means وجود دارد؛ یعنی انجام متوالی دو گام به‌روز کردن K زیرفضا و تخصیص هر داده به نزدیک‌ترین زیرفضا. بعبارت دقیق‌تر، ابتدا K زیرفضای s بُعدیِ متعامد یکّه بصورت تصادفی انتخاب می‌شود. اگر این زیرفضاها را با ماتریس‌های $\mathbf{D}_r \in \mathbb{R}^{n\times s},\ r = ۱,\dots,K$ نشان دهیم، هر داده‌ی نوعیِ $\mathbf{y}$ براساس معیار زیر به نزدیک‌ترین زیرفضا اختصاص داده می‌شود:

$$\min_r \|\mathbf{y} - \mathbf{D}_r\mathbf{D}_r^T\mathbf{y}\|_۲^۲. \qquad (۴۴–۶)$$

بعد از این که همه‌ی داده‌ها به یک زیرفضا اختصاص داده شدند، هر پایه با اعمال SVD به داده‌های موجود در آن و سپس انتخاب اولین s مؤلفه‌ی اساسی، به‌روز می‌شود. تفاوت اصلی الگوریتم SSF با K-subspace در این است که در الگوریتم SSF زیرفضاها بصورت متوالی پیدا می‌شوند و در نتیجه، نیازی به دانستن تعداد زیرفضاها نیست.

۶–۸ شبیه‌سازی

در این قسمت عملکرد الگوریتم‌های پیشنهادی را با انجام چندین شبیه‌سازی ارزیابی می‌کنیم. به طور کلّی دو نوع آزمایش برای بررسی کارآئی یک الگوریتم آموزش دیکشنری انجام می‌شود؛ یکی روی داده‌های ساختگی یا مصنوعی و دیگری روی داده‌های واقعی یا طبیعی. در آزمایش اول، یک دیکشنری با ابعاد مشخص به طریقی که

در ادامه می‌گوئیم ساخته می‌شود. سپس با استفاده از آن تعدادی داده‌ی آموزشی تولید می‌شود. هر داده با استفاده از ترکیب خطی تعداد مشخصی اتم ساخته می‌شود. در نهایت، این داده‌ها با میزان معینی نویز جمع شده و به الگوریتم داده می‌شود. کلّیت این آزمایش شبیه همان آزمایشی است که در [۱] انجام شده است. یک الگوریتم کاراً بعنوان یک پیش‌نیاز باید قادر باشد دیکشنری مولّد[۱] را با دقت خوبی بازسازی کند. مطابق مطالب فصل ۳، نقش اصلی آموزش دیکشنری برای یک دسته داده‌ی آموزشی، استخراج ویژگی‌های برجسته‌ی آن داده‌ها است. در نتیجه، الگوریتم مورد آزمایش نیز باید بتواند درصد زیادی از ویژگی‌های برجسته‌ی داده‌ها را، که همان اتم‌های مولّد هستند، استخراج کند.

همانطور که در فصل ۳ توضیح دادیم، بسیاری از سیگنال‌های طبیعی بر خلاف بُعد ظاهری بالای خود، بُعد ذاتیِ به مراتب کمتری دارند. بعنوان مثال، تعداد ویژگی‌ها یا همان اتم‌هائی که هر تصویر (یا یک بلوک از آن) را توصیف می‌کند نسبت به تعداد پیکسل‌های آن بسیار کمتر است. در این دسته از داده‌ها امّا تعداد واقعی اتم‌های مولّد از قبل معلوم نیست. همه‌ی الگوریتم‌های موجود (تا جائیکه نگارنده اطلاع دارد) برای آموزش دیکشنری یک تعدادِ مشخص اتم فرض می‌کنند. این که الگوریتم‌های فعلی از ابتدا تعداد اتم‌ها را معلوم فرض کرده و سپس این اتم‌ها را برای استخراج ویژگی از داده‌های آموزشی بهینه می‌کنند، شاید بزرگترین مشکل آن‌ها است. ایده‌ی بهتر امّا می‌تواند به این صورت باشد که این اتم‌ها یا ویژگی‌ها را یکی یکی از داده‌ها استخراج کنیم. این کار بسیار شبیه عملکرد الگوریتم OMP برای انتخاب اتم به منظور نمایش یک سیگنال است. در OMP، ویژگی‌های برجسته‌ی یک سیگنال یکی یکی از یک مجموعه‌ی مشخص، که همان دیکشنری است، انتخاب می‌شود. این ایده بعنوان یک راه‌حل خوب برای غلبه بر مشکل مذکور به عنوان کارهای آینده تحت بررسی قرار خواهد گرفت[۲]. به هر حال، آزمایش دوم روی یک سری داده‌ی طبیعی انجام شده و با فرض یک تعداد مشخص اتم، توانائی الگوریتم در استخراج برجسته‌ترین ویژگی‌های داده‌ها ارزیابی می‌شود. در این بخش، ما آزمایش دوم را هم روی سیگنال‌های یک بُعدی، با انتخاب بلوک‌هائی از یک سیگنال AR[۳] و هم روی تصاویر، در کاربرد نویززدائی، انجام می‌دهیم.

[۱] Generating Dictionary

[۲] اخیراً ایده‌ای مشابه در [۱۲] پیشنهاد شده است که در آن، اتم‌های دیکشنری نهائی با رویکردی حریص از میان چندین پایه‌ی کاندید انتخاب می‌شوند. هدف اصلی این مقاله امّا پُرکردن فاصله‌ی موجود بین «طراحی» و «آموزش» دیکشنری بوده و با آنچه که ما به دنبالش هستیم تفاوت دارد.

[۳] Auto-Regressive

در تمام آزمایش‌های این فصل از بسته‌های KSVD-Box v13 و OMP-Box v10[1] برای پیاده‌سازی K-SVD و OMP استفاده کرده‌ایم. این آزمایش‌ها روی سیستمی دارای پردازنده‌ی Intel Core i7، سرعت پردازش GHz ۳٫۸، حافظه‌ی GB ۸ و تحت نرم‌افزار MATLAB نسخه‌ی R2010b انجام شده است. در ادامه، شیوه‌ی انجام هر آزمایش را با جزئیات بیش‌تری بررسی می‌کنیم.

۶-۸-۱ داده‌های مصنوعی

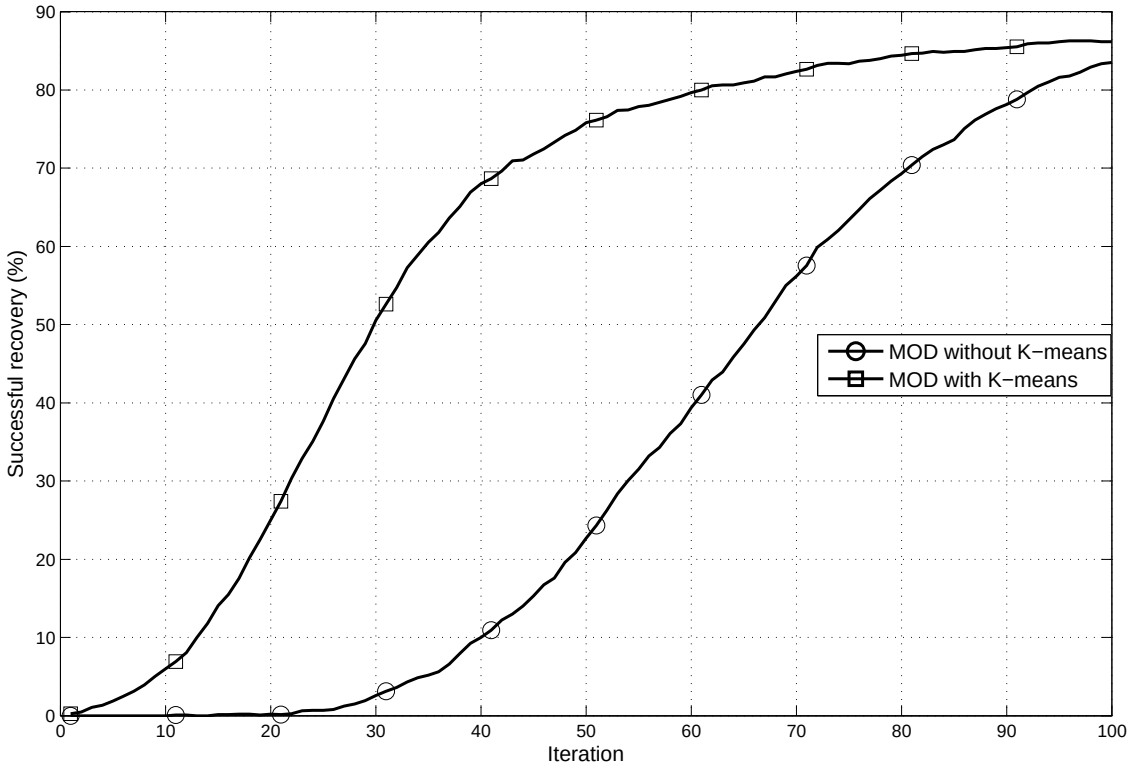
ابعاد دیکشنری در این آزمایش برابر با ۵۰×۲۰ است. درایه‌های این دیکشنری مستقل از یکدیگر و از توزیع گوسی با میانگین صفر و واریانس یک انتخاب شده و در انتها ستون‌های آن نُرمالیزه می‌شود. تعداد ۲۰۰۰ داده‌ی آموزشی، هر یک با استفاده از یک ترکیب خطی از s اتم ساخته می‌شود. مقدار درایه‌های غیرصفر مستقل از هم و از توزیع گوسی با میانگین صفر و واریانس یک و موقعیت آن‌ها نیز بصورت تصادفی با توزیع یکنواخت انتخاب می‌شود. مقدار s از ۳ تا ۶ متغیّر است. در انتها، نویزی گوسی با میانگین صفر و وارایانس متغیّر به داده‌ها اضافه می‌شود. سطح نویز را برابر با ۱۰ dB، ۲۰ dB، ۳۰ dB و ۱۰۰ dB (معادل با حالت بدون نویز) می‌گیریم. هر آزمایش متناظر با یک مقدار برای s و یک سطح نویز مشخص است.

برای بررسی عملکرد الگوریتم‌های مختلف، ماتریس $\mathbf{Y}$ را به تمام الگوریتم‌ها می‌دهیم. هر الگوریتم نهایتاً یک دیکشنری به ما می‌دهد که بر حسب فاصله‌ی آن تا دیکشنری اصلی (از نظر میزان شباهت ستون‌های آن‌ها؛ البته با درنظر گرفتن جابجا شدن ستون‌ها) درصد بازیابی درست اتم‌های دیکشنری را برای آن الگوریتم محاسبه می‌کنیم. تعداد ۱۰۰ تناوب بین دو گام آموزش دیکشنری انجام می‌شود. می‌گوئیم یک الگوریتم با موفقیت اتم $\mathbf{d}$ را بازیابی کرده است هرگاه $|\mathbf{d}^T\hat{\mathbf{d}}| > ۰٫۹۹$ که $\hat{\mathbf{d}}$ شبیه‌ترین اتم به $\mathbf{d}$ در بین اتم‌های دیکشنری خروجی الگوریتم است. نهایتاً موفقیت هر الگوریتم را با استفاده از درصد بازیابی درست اتم‌ها اندازه می‌گیریم. برای کاهش اثر تصادفی، ۵۰ بار هر آزمایش را تکرار کرده و روی نتایج میانگین می‌گیریم.

در تقریباً تمامی الگوریتم‌های آموزش دیکشنری (تا جائی که نگارنده اطلاع دارد) دیکشنری اولیه‌ای که بعنوان شروع الگوریتم بهینه‌سازی نوبتی استفاده می‌شود، با انتخاب تصادفی K داده آموزشی و سپس نُرمالیزه کردن آنها بدست می‌آید[2]. امّا همانطور که قبلا گفته شد، مسأله‌ی آموزش دیکشنری نسبت به دو متغیّر آن غیر محدّب

[1] `http://www.cs.technion.ac.il/~ronrubin/software.html`

[2] البته در کاربردهائی که کلاس داده‌ها کاملا مشخص است، از یک دیکشنری مناسب برای آن کلاس استفاده می‌شود؛ مانند دیکشنری فوق

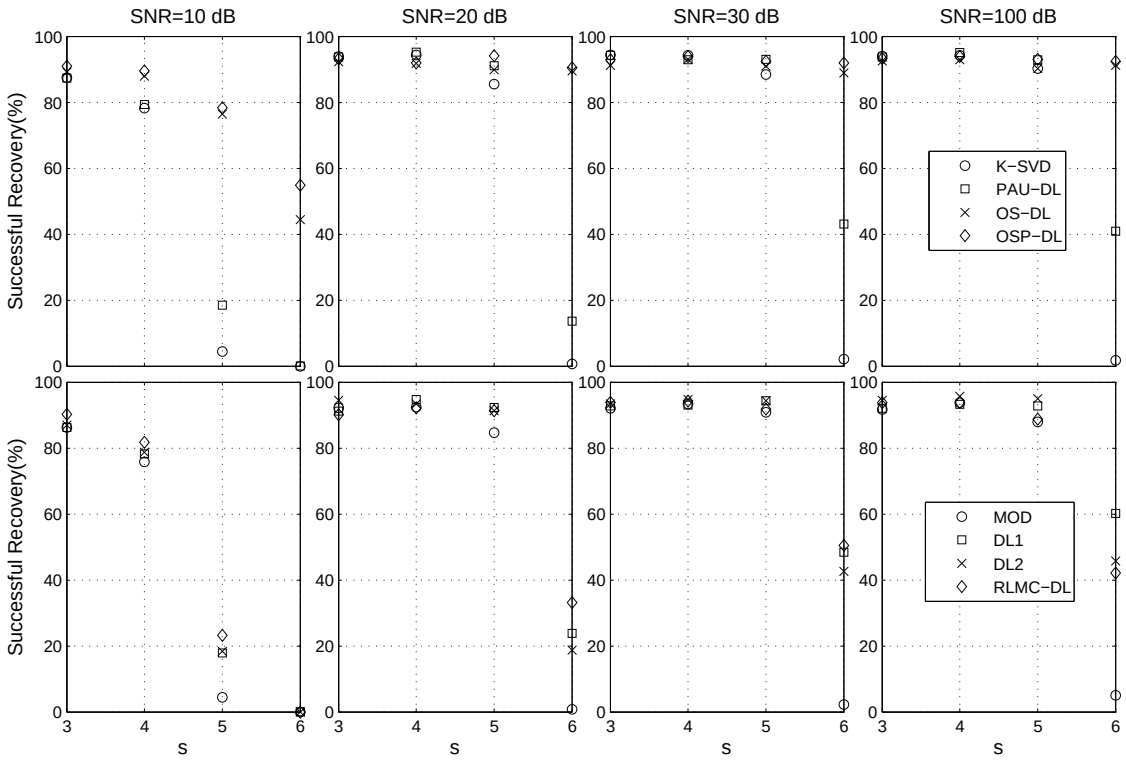


شکل ۶–۸: تاثیر مقداردهی اولیه‌ی دیکشنری روی همگرائی الگوریتم MOD. این شکل درصد بازیابی درست دیکشنری را برحسب شماره‌ی تکرار ، با استفاده از الگوریتم MOD و برای دو حالت مقداردهی با K-means و مقداردهی تصادفی نشان می‌دهد.

بوده و بنابراین در حالت کلّی تعداد زیادی مینیمم محلّی دارد. در نتیجه، مقداردهی اولیه الگوریتم از اهمّیت بالائی برخوردار است؛ چرا که اگر دیکشنری اولیه هوشمندانه انتخاب نشود (مشابه الگوریتم‌های فعلی)، ممکن است سرعت و کیفیت همگرائی الگوریتم افت پیدا کند.

برای رفع این مشکل، یک گزینه‌ی مناسب، استفاده از الگوریتم معروف خوشه‌بندی K-means است؛ به این ترتیب که دیکشنری اولیه را برابر با مرکز خوشه‌های نهائی الگوریتم K-means انتخاب می‌کنیم. به بیان دقیق‌تر، ماتریس داده‌ها را به همراه تعداد اتم‌های دیکشنری به الگوریتم K-means می‌دهیم، و خروجی این الگوریتم را که شامل K مرکز خوشه است، بعد از نُرمالیزه کردن، بعنوان دیکشنری اولیه انتخاب می‌کنیم. با این کار، ماتریس ضرایب $\mathbf{X}$ اولیه برخلاف الگوریتم‌های قبلی، تُنُک خواهد بود. با توجه به این که امروزه الگوریتم‌های بسیار کارآئی برای خوشه‌بندی وجود دارد [۳۷]، بنابراین حجم محاسبات آن بسیار ناچیز خواهد بود؛ چیزی که در شبیه‌سازی‌ها

کامل DCT در کلاس تصاویر طبیعی.

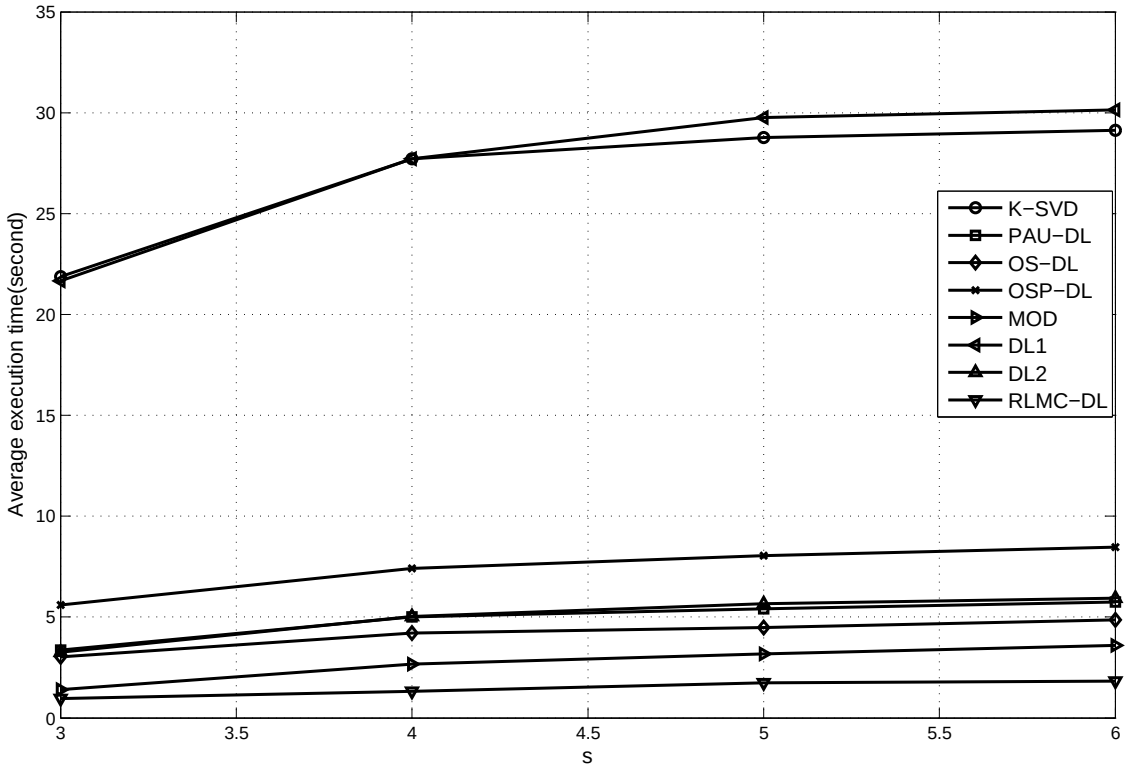


شکل ۶–۹: نتایج مربوط به آزمایش روی داده‌های مصنوعی. هر ردیف متناظر با ۴ الگوریتم بوده که در شکل مشخص شده‌اند. هر ستون نیز متناظر با یک سطح نویز مشخص است.

هم مشاهده شد. بعلاوه، در کاربردهائی که تعداد داده‌های آموزشی خیلی زیاد است، می‌توانیم به جای کلّ ماتریس $\mathbf{Y}$ ، که تعداد ستونهای آن خیلی زیاد است، تنها درصدی از ستونهای آن را به صورت تصادفی انتخاب کرده و به الگوریتم K-means بدهیم. برای مشاهده‌ی تاثیر این مقداردهی، شکل ۶–۸ را ببینید. این شکل درصد بازیابی درست دیکشنری را برحسب شماره‌ی تکرار، برای الگوریتم MOD و دو حالت مقداردهی با K-means و مقداردهی تصادفی نشان می‌دهد. سطح نویز برابر با ۲۰ dB بوده و $s = ۵$ انتخاب شده است.

نکته‌ی دیگر این که به دلیل ماهیت غیرمحدّب مسأله، در برخی تکرارها ممکن است الگوریتم در یک مینیمم محلّی گرفتار شده و چند تناوب داخل آن نوسان کند. به همین دلیل، استفاده از قید $\|\mathbf{D}^{(k+۱)} - \mathbf{D}^{(k)}\|_F \leq \epsilon$ برای توقف الگوریتم می‌تواند زمان اجرا را خیلی بالا ببرد. در این آزمایش‌ها مشابه مراجع موجود، شرط توقف را رسیدن به یک تعداد تکرار (یا همان تناوب) مشخص قرار می‌دهیم. برای این منظور، تعداد ۱۰۰ تناوب بین دو گام آموزش دیکشنری انجام می‌دهیم.

نتایج این آزمایش برای ۸ الگوریتم در شکل ۶–۹ آورده شده است. شکل‌های ردیف بالائی مربوط به



شکل ۶–۱۰: زمان متوسط اجرای الگوریتم‌های مختلف برحسب s در آزمایش روی داده‌های مصنوعی.

الگوریتم‌های K-SVD، PAU-DL، OS-DL و OSP-DL بوده که همگی در گام به‌روز کردن دیکشنری، اتم‌ها را «یکی یکی» به‌روز می‌کنند. شکل‌های ردیف پائینی هم مربوط به الگوریتم‌های MOD، DL1، DL2 و RLMC-DL (با $\lambda = ۲$) است که همگی در گام به‌روز کردن دیکشنری، همه‌ی اتم‌ها را «یکجا» به‌روز می‌کنند. شکل‌های هر ستون متناظر با یک سطح نویز مشخص است. زمان متوسط اجرای هر الگوریتم برحسب s در شکل ۶–۱۰ رسم شده است. با بررسی نتایج بدست آمده، مشاهدات زیر را نتیجه می‌گیریم:

۱. الگوریتم PAU-DL نسبت به K-SVD در بازیابی درست اتم‌ها عملکرد بهتری دارد. این موضوع بخصوص برای $s = ۵$ و $s = ۶$ مشهود است. بعلاوه، PAU-DL از نظر زمان اجرا حدود ۵ برابر سریع‌تر از K-SVD است.

۲. الگوریتم‌های OS-DL و OSP-DL نسبت به دو الگوریتم هم‌گروه خود، یعنی K-SVD و PAU-DL، و بخصوص برای $s = ۵$ و $s = ۶$ عملکرد بهتری دارند. همچنین OSP-DL نسبت به OS-DL اندکی بهتر عمل کرده است امّا با این وجود، زمان اجرای آن بیشتر است.

۳. عملکرد دو الگوریتم MOD و K-SVD همانطور که در [۵۳] نیز ذکر شده، بسیار شبیه هم است.

۴. دو الگوریتم DL1 و DL2 بخصوص برای s های کمتر از ۶ عملکرد بسیار مشابهی دارند. بعلاوه، زمان اجرای DL2 حدود ۵ برابر کمتر از DL1 است. هم‌چنین، الگوریتم‌های DL2 و PAU-DL بسیار شبیه هم عمل کرده‌اند. توجه کنید که اگرچه DL2 در واقع تقریبی از DL1 است، علّت عملکرد مشابه این دو الگوریتم احتمالاً به همان دلیلی است که در بخش ۶–۲ اشاره کردیم؛ یعنی ممکن است در چند تکرار، ماتریس $\mathbf{D}_r$ در معادله‌ی (۶–۵) بدحالت شود.

۵. الگوریتم RLMC-DL نسبت به الگوریتم مشابه خود، یعنی MOD، و بخصوص برای $s = ۵$ و $s = ۶$ عملکرد بهتری دارد. هم‌چنین، با وجود این که در الگوریتم RLMC-DL برخلاف MOD ماتریس $\mathbf{D}^T\mathbf{D}$ در هر تناوب محاسبه می‌شود، امّا زمان اجرای RLMC-DL نسبت به MOD کمتر است!

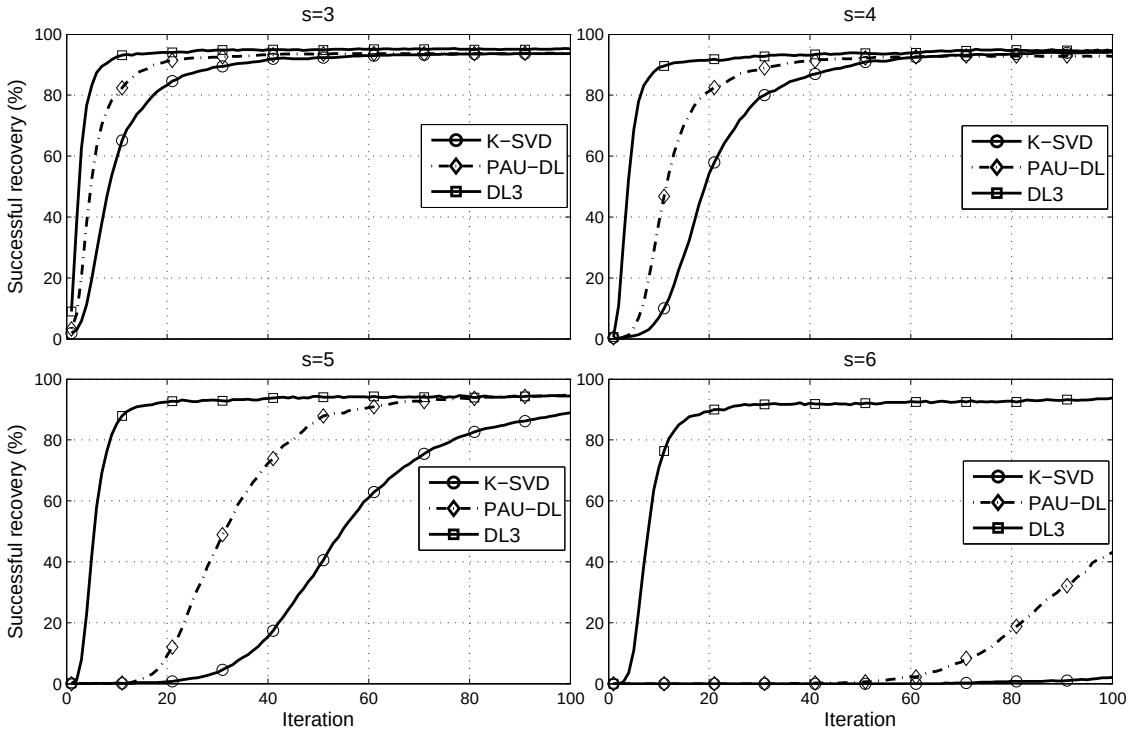
عملکرد الگوریتم DL3، که ساختار آن با همه‌ی الگوریتم‌های فوق متفاوت است، از نظر درصد بازیابی درست اتم‌ها و بخصوص سرعت همگرائی، نسبت به بقیه‌ی الگوریتم‌ها بهترین است. برای s برابر با ۳ تا ۶ و سطح نویز ۳۰ dB، نمودارهای همگرائی سه الگوریتم K-SVD، PAU-DL و DL3 در شکل ۶–۱۱ رسم شده است. با دقت در این شکل می‌توان گفت که همگرائی الگوریتم PAU-DL نسبت به K-SVD سریع‌تر بوده و بعلاوه، الگوریتم DL3 همگرائی بسیار بهتری دارد؛ به طوری که بعد از حدود ۱۰ تکرار و مستقل از مقدار s همگرا شده است.

۶–۸–۲ سیگنال AR(1)

این آزمایش همانی است که در [۵۳] به عنوان آزمایش روی داده‌های واقعی انجام شده است؛ با این تفاوت که در اینجا اندازه‌ی دیکشنری و تعداد داده‌ها متفاوت است. داده‌های آموزشی برای این آزمایش بصورت بلوک‌های از یک سیگنال AR(1) انتخاب می‌شوند. این سیگنال بصورت زیر تولید می‌شود:

$$v(t) = ۰/۹۵v(t-۱) + e(t), \qquad (۴۵–۶)$$

که در آن $e(t)$ نویزی گوسی با میانگین صفر و واریانس یک است. تعداد ۲۰۰۰ داده‌ی آموزشی با انتخاب بلوک‌هائی به طول ۲۰ از این سیگنال تولید می‌شود. تعداد اتم‌های دیکشنری برابر با $m = ۴۰$ انتخاب می‌شود. در گام نمایش تُنُک، حداکثر تعداد اتم‌های مورد استفاده برای نمایش هر داده را برابر با ۵ می‌گیریم. در هر الگوریتم، برای آموزش دیکشنری تعداد ۱۰۰ تکرار انجام می‌شود. دقت کنید که همانطور که در [۵۳] نیز گفته شده است، هیچ ساختار تُنُکِ



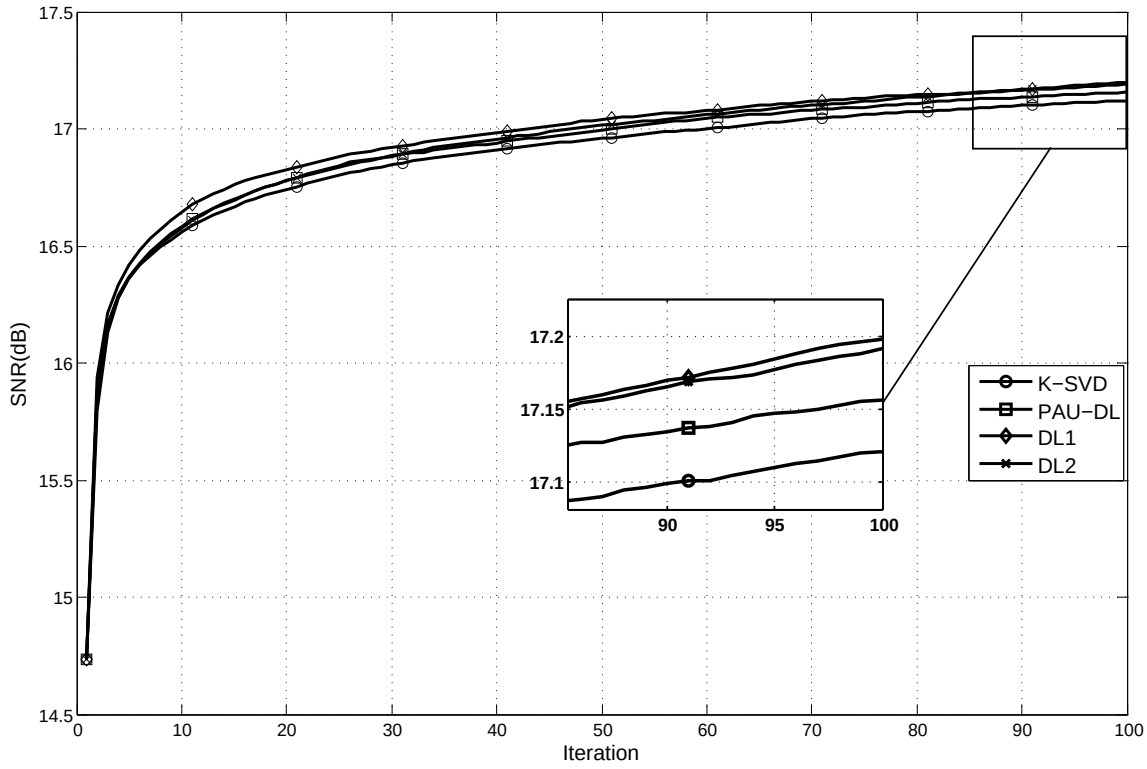
شکل ۶–۱۱: نمودارهای همگرائی سه الگوریتم K-SVD، PAU-DL و DL3 برای مقادیر مختلف s و سطح نویز ۳۰ dB.

معینی بر این بلوک‌ها حاکم نیست و هدف فقط حداقل کردن خطای نمایش تُنُک داده‌ها است. نسبت سیگنال به نویز در هر تناوب از آموزش دیکشنری را، مطابق با آنچه در [۵۳] گفته شده است، بصورت زیر تعریف می‌کنیم:

$$\mathrm{SNR} = ۱۰ \log \frac{\sum_i \|\mathbf{y}_i\|_۲^۲}{\sum_i \|\mathbf{y}_i - \mathbf{D}\mathbf{x}_i\|_۲^۲}. \qquad (۴۶–۶)$$

الگوریتمی که مقدار SNR خروجی آن بیشتر باشد به این معنی است که توانسته ویژگی‌های برجسته‌تری از داده‌ها را استخراج کند. این آزمایش را برای الگوریتم‌های مختلف ۵۰ بار تکرار کرده و روی نتایج خروجی میانگین می‌گیریم. مشابه [۵۳]، برای بررسی کارآئی الگوریتم‌های مختلف، مقدار SNR را برحسب شماره‌ی تکرار رسم می‌کنیم. نتایج مربوط به چهار الگوریتم K-SVD، PAU-DL، DL1 و DL2 در شکل ۶–۱۲ رسم شده است. همانطور که از این شکل پیدا است، عملکرد الگوریتم DL1 مشابه DL2 بوده و نسبت به دو الگوریتم دیگر بهتر است. هم‌چنین، PAU-DL بهتر از K-SVD عمل کرده است.

نتایج مربوط به دو الگوریتم OSP-DL و DL3 (با $\lambda = ۰/۰۱$) در شکل ۶–۱۳ آورده شده است. برای مقایسه، بهترین نتیجه در بین چهار الگوریتم قبلی، که مربوط به الگوریتم DL1 است، نیز در این شکل رسم شده



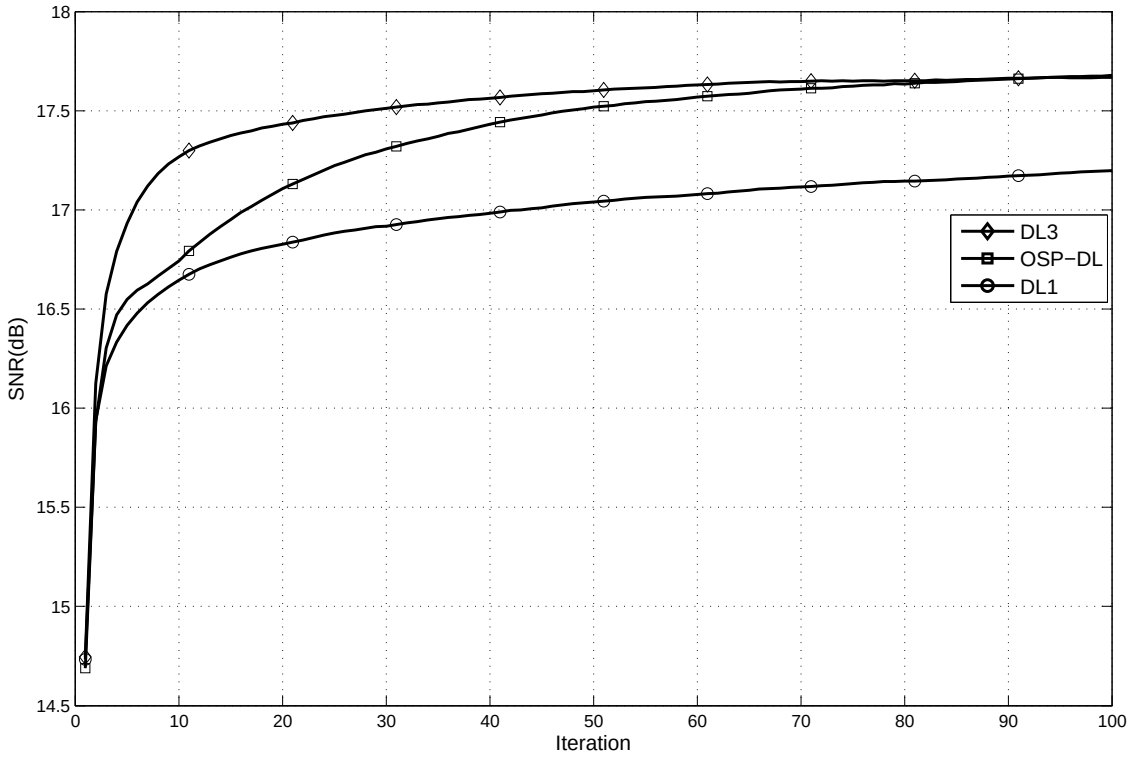
شکل ۶-۱۲: نمودار SNR برحسب شماره‌ی تکرار برای چهار الگوریتم K-SVD، PAU-DL، DL1 و DL2 مربوط به آزمایش روی سیگنال AR(1).

است. این شکل نشان می‌دهد که نتایج دو الگوریتم OSP-DL و DL3 شبیه هم بوده و نسبت به چهار الگوریتم قبلی بهتر است.

مقدار SNR خروجی و زمان اجرای الگوریتم OSP-DL به‌ازای چند مقدار مختلف پارامتر λ (ضریب رگولاریزیشن) در شکل ۶-۱۴ رسم شده است. بهترین مقدار SNR برای $\lambda \simeq ۰/۰۱$ بدست می‌آید. به تغییرات زمان اجرای الگوریتم برحسب مقادیر λ دقت کنید. هرچه مقدار λ کوچک‌تر باشد، زمان اجرای الگوریتم بیش‌تر است. توجیه این اتفاق به این ترتیب است که هرچه مقدار λ کوچک‌تر باشد، میزان انقباض ضرایب در رابطه‌ی (۶-۳۲) کم‌تر بوده و در نتیجه، نمایه‌ی مورد نظر حاوی تعداد بیش‌تری درایه‌ی غیرصفر خواهد بود. بنابراین، محاسبه‌ی مقدار به‌روز شده‌ی اتمِ متناظر حجم محاسبات بالاتری می‌طلبد. مشابه این موضوع قبلاً در [۳۵] برای حلّ مسأله‌ی غیر مقیّد زیر مورد بررسی قرار گرفته است:

$$\min_{\mathbf{x}} f(\mathbf{x}) = \frac{۱}{۲}\|\mathbf{y} - \mathbf{D}\mathbf{x}\|_۲^۲ + \lambda\|\mathbf{x}\|_۱. \qquad (۶-۴۷)$$

به عبارت دیگر، در [۳۵] نشان داده شده است که حلّ مسأله‌ی فوق برای مقادیر بزرگ‌تر پارامتر λ (متناظر با

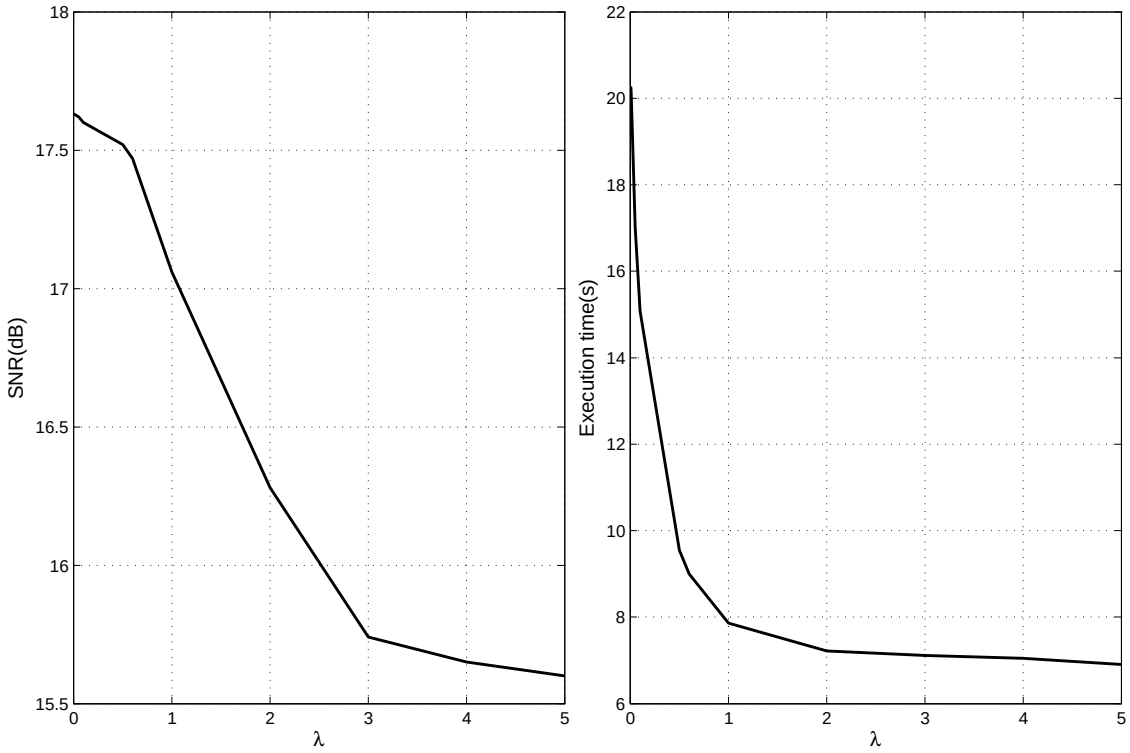


شکل ۶–۱۳: نمودار SNR برحسب شماره‌ی تکرار برای سه الگوریتم DL3، OSP-DL و DL1 مربوط به آزمایش روی سیگنال AR(1).

جواب‌های تُنُک‌تر) زمان کم‌تری صرف می‌کند. ایده‌ی [۳۵] سپس حلّ این مسأله (با استفاده از الگوریتم‌های IST) برای یک دنباله‌ی کاهشی از مقادیر λ و بعلاوه، استفاده از جواب نهائی مسأله‌ی متناظر با یک مقدارِ λ بعنوان جواب اولیه برای حلّ مسأله‌ی متناظر با مقدار بعدیِ λ است[۱]. آزمایشی که در [۳۵] انجام شده است نشان می‌دهد که استفاده از این ایده برای حلّ (۶–۴۷) تأثیر زیادی در سرعت همگرائی و در نتیجه کاهش زمان اجرای الگوریتم دارد. با این توضیح، می‌توان از این ایده در OSP-DL و DL3 استفاده کرد. ما بررسی این موضوع را به آینده موکول می‌کنیم.

برای بررسی کارآئی الگوریتم RLMC-DL و مشاهده‌ی تأثیر پارامتر λ در این الگوریتم، این آزمایش را برای نُه مقدار $\lambda = ۱, ۱۰, ۲۰, \ldots, ۸۰$ انجام داده‌ایم. نتایج این آزمایش در شکل ۶–۱۵ رسم شده است. برای مقایسه، نتیجه‌ی مربوط به الگوریتم MOD نیز آورده شده است. با دقت در این شکل مشخص است که عملکرد الگوریتم RLMC-DL بهتر از MOD بوده و بعلاوه، با افزایش مقدار پارامتر λ نتایج بهتر می‌شود.

[۱] به شباهت این ایده با روش GNC توجه کنید.



شکل ۶-۱۴: نمودار SNR خروجی و زمان اجرای الگوریتم OSP-DL برحسب مقادیر مختلف پارامتر λ. نتایج مربوط به آزمایش روی سیگنال AR(1) است.

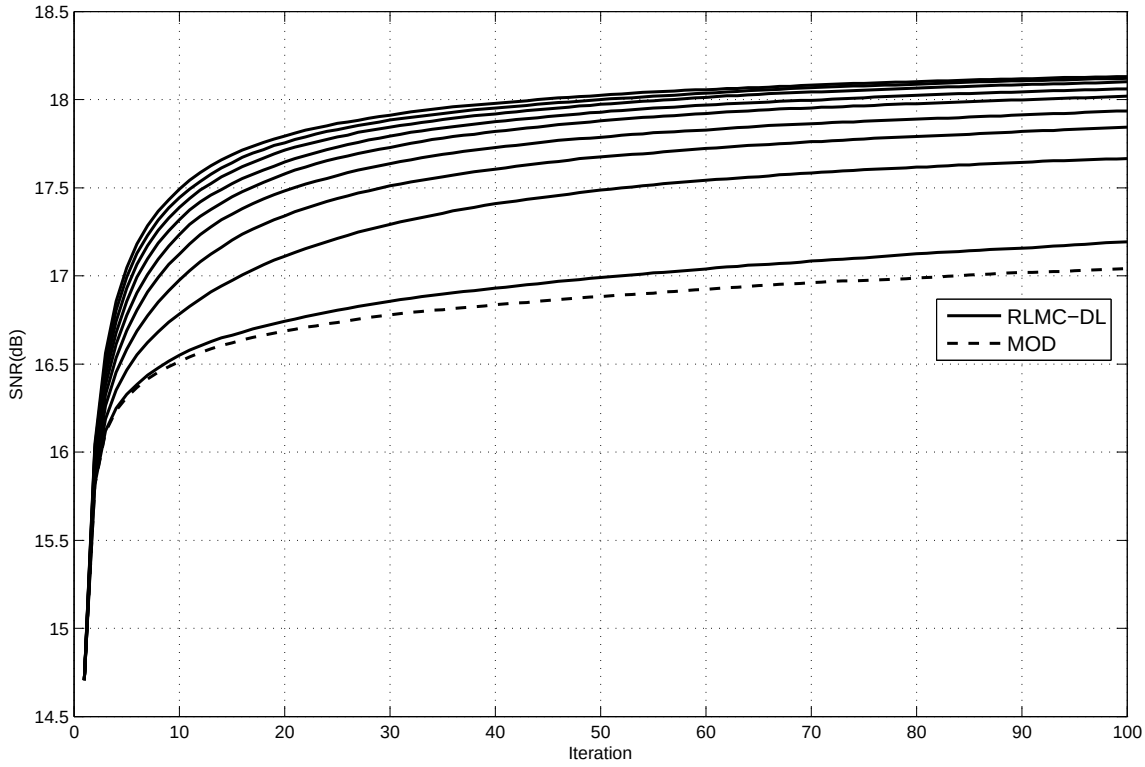
برای مقایسه‌ی ساده‌تر، مقادیر نهائی SNR برای الگوریتم‌های مختلف به همراه زمان متوسط اجرای آن‌ها در جدول ۶-۱ خلاصه شده است.

جدول ۶-۱: SNR خروجی و زمان اجرای الگوریتم‌های مختلف مربوط به آزمایش روی سیگنال AR(1).

Algorithm	K-SVD	PAU-DL	DL1	DL2	RLMC-DL	MOD	DL3	OSP-DL
SNR(dB)	۱۷٫۱۲	۱۷٫۱۶	۱۷٫۲	۱۷٫۲	۱۸٫۱۶	۱۷٫۰۳	۱۷٫۶۳	۱۷٫۶۳
Time(s)	۱۴٫۵۷	۲٫۵۳	۱۹٫۸۳	۲٫۴	۱٫۱۵	۱٫۱۷	۳۵	۱۲٫۹۳

۶-۸-۳ نویززدائی از تصاویر

نویززدائی از تصاویر با استفاده از نمایش تُنُک را در فصل ۴ بررسی کردیم. در این رهیافت، ابتدا یک دیکشنری با استفاده از تعدادی داده‌ی آموزشی به کمک یکی از الگوریتم‌های موجود آموزش می‌بینند. سپس، بلوک‌های تصویر نویزی با استفاده از این دیکشنری بی‌نویز می‌شوند. همانطور که در این فصل گفتیم و در [۲۵] گزارش شده است، استفاده از بلوک‌های خودِ تصویر نویزی به عنوان داده‌ی آموزشی نتایج بهتری نسبت به استفاده از بلوک‌های



شکل ۶–۱۵: نمودار SNR برحسب شماره‌ی تکرار برای دو الگوریتم RLMC-DL و MOD مربوط به آزمایش روی سیگنال AR(1). نمودارهای پیوسته از پائین به بالا به ترتیب متناظر با مقادیر $\lambda = ۱$، $\lambda = ۱۰$، $\lambda = ۲۰$ تا $\lambda = ۸۰$ است.

تعدادی تصویر با کیفیت بدست می‌دهد.

اکثر الگوریتم‌های موجود برای آموزش دیکشنری مبتنی بر پردازش تمام داده‌های آموزشی در گام به‌روز کردن دیکشنری هستند. مشکل عمده‌ی این دسته از الگوریتم‌ها، بخصوص K-SVD، این است که به دلیل محاسبات زیاد، روی تعداد داده‌های آموزشی محدودیت دارند. بعنوان مثال، تعداد کلّ بلوک‌های با ابعاد ۸×۸ از یک تصویر با اندازه‌ی ۲۵۶×۲۵۶ برابر است با $۶۲۰۰۱ = {}^{۲}(۱ + ۸ - ۲۵۶)$. پردازش این تعداد بلوک، حجم محاسبات بالائی می‌طلبد. دسته‌ی دوم الگوریتم‌های آموزش دیکشنری، همانطوری که در فصل ۳ گفتیم، مبتنی بر استفاده از ایده‌ی «آموزش آنلاین» هستند که در آن در گام به‌روز کردن دیکشنری، هر بار تنها یک داده‌ی آموزشی پردازش می‌شود. دقت کنید که در این الگوریتم‌ها در هر تناوب از «تمام» داده‌های آموزشی استفاده می‌شود؛ ولی چون داده‌ها یکی یکی پردازش می‌شوند، بنابراین قادر هستند هر تعداد داده را پردازش کنند. بعنوان مثال، در [۴۴] یک آزمایش روی تصویری با ۱۲ میلیون پیکسل انجام شده است؛ چیزی که برای یک الگوریتم مبتنی بر پردازش یک‌جای داده‌ها عملاً غیرممکن است. بنابراین، توسعه‌ی الگوریتم‌های آنلاین برای آموزش دیکشنری در کاربردهای با ابعاد بالا

شکل ۶–۱۶: تصاویر تست مورد استفاده برای نویززدائی. به ترتیب از راست به چپ: Boat، House، Lena و Barbara.

موضوعی بسیار خوب برای کارهای آینده است.

ما در این قسمت برای ارزیابی عملکرد الگوریتم‌های پیشنهادی از ۴ تصویر تست، همگی با ابعاد ۲۵۶×۲۵۶، استفاده می‌کنیم. این تصاویر در شکل ۶–۱۶ آورده شده‌اند. بنا به دلیل ذکر شده در پاراگراف قبل، ما تنها از ۳۰۰۰۰ بلوکِ ۸×۸ از تصویر نویزی به عنوان داده‌ی آموزشی استفاده می‌کنیم. این بلوک‌ها مطابق [۲۵] بصورت تصادفی انتخاب می‌شوند[۱]. در گام نمایش تُنُک، مطابق با آنچه در [۲۵] پیشنهاد شده است، الگوریتم OMP تا جائی برای نمایش هر بلوک اتم انتخاب می‌کند که خطای نمایشِ آن به زیر $\epsilon = ۱٫۱۵\sqrt{n}\sigma$ برسد. معیار ارزیابی نتیجه‌ی خروجی یک الگوریتم، نسبت حداکثر توان سیگنال به توان نویز بوده که بصورت زیر تعریف می‌شود:

$$\mathrm{PSNR} = ۱۰ \log \frac{۲۵۵^۲}{\frac{۱}{N}\sum_i |\hat{y}_i - y_i|^۲}, \qquad (۴۸–۶)$$

که در آن y_i و $\hat{y}_i$ به ترتیب، پیکسل iاُم از تصویر نویزی و تصویر خروجی الگوریتم بوده و N تعداد کلّ پیکسل‌های تصویر است.

در این قسمت، عملکرد سه الگوریتم K-SVD، PAU-DL و RLMC را با هم مقایسه می‌کنیم. هر آزمایش متناظر با یک تصویر و با یک سطح نویز مشخص است. برای کاهش اثر تصادفی ناشی از نویز، هر آزمایش ۵ بار تکرار شده و روی نتایج میانگین‌گیری انجام می‌شود. دیکشنری اولیه را مشابه [۲۵] برابر با دیکشنری DCT فوق کامل می‌گیریم. به این ترتیب، الگوریتم‌های مورد بحث با شروع از این دیکشنری، در هر تناوب اتم‌های آن را در جهتِ نمایش بهتر بلوک‌های تصویر تغییر می‌دهند. بنابراین، برای این که ببینیم نتیجه‌ی نهائی نویززدائی با دیکشنری آموزش دیده چه بهبودی نسبت به استفاده‌ی از دیکشنری اولیه (همان دیکشنری گام نخست) داشته است، نتایج مربوط به دیکشنری فوق کامل DCT را نیز به همراه نتایج الگوریتم‌های مذکور در جدول ۶–۲ آورده‌ایم.

[۱] این طرز انتخاب خود جای بحث دارد؛ چرا که نوع بلوک‌ها از این نظر که حاوی ویژگی برجسته‌ای هستند یا نه، نقش مهمی در آموزش دیکشنری یا استخراج ویژگی دارد. بعبارت دیگر، «غنای» داده‌های آموزشی برای عملکرد یک الگوریتم معیار مهمی است. ما در اینجا این موضوع را بررسی نکرده و آن را به آینده موکول می‌کنیم.

شکل ۶–۱۷: تصویر حذف نویز شده‌ی Lena توسط الگوریتم PAU-DL. از چپ به راست: تصویر بدون نویز، تصویر نویزی (PSNR = ۲۰٫۱۷ dB) و تصویر حذف نویز شده (PSNR = ۳۰٫۱۱ dB).

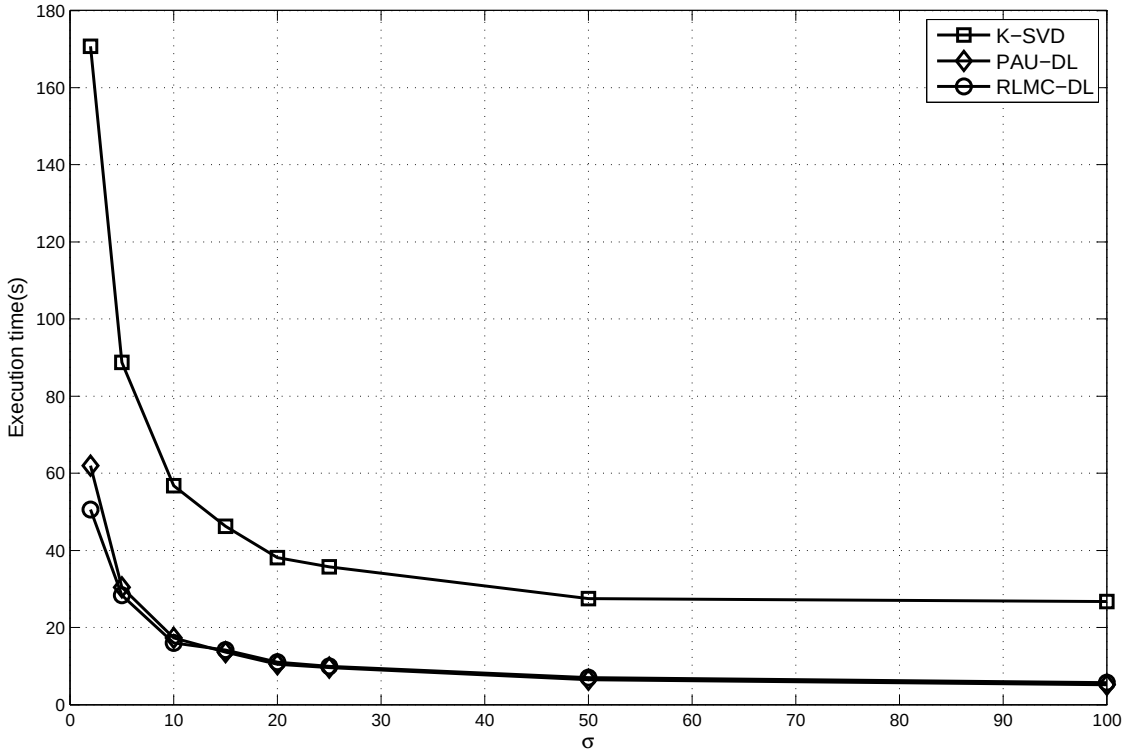
زمان متوسط اجرای الگوریتم‌های مختلف در شکل ۶–۱۸ آمده است.

با دقت در نتایج بدست آمده مشاهده می‌شود که الگوریتم PAU-DL هم از نظر مقدار متوسط PSNR و هم از نظر زمان اجرا، بهترین عملکرد را نسبت به بقیه‌ی الگوریتم‌ها دارد. در نتیجه، این آزمایش نیز بیانگر این است که با توجه به زمان اجرای بالای K-SVD، عملاً می‌توان از الگوریتم PAU-DL، که نسبت به آن چند برابر سریع‌تر بوده و نتایج خروجی آن هم به طور متوسط بهتر است، استفاده کرد. برای بررسی کیفیت تصاویر حذف نویز شده از نظر دیداری، تصویر حذف نویز شده Lena توسط الگوریتم PAU-DL به‌همراه نُسخه‌های تمیز و نویزی آن در شکل ۶–۱۷ آورده شده است. همانطور که مشاهده می‌شود، الگوریتم PAU-DL علاوه بر تضعیف نویز، جزئیات تصویر اصلی را نیز تقریباً حفظ کرده است.

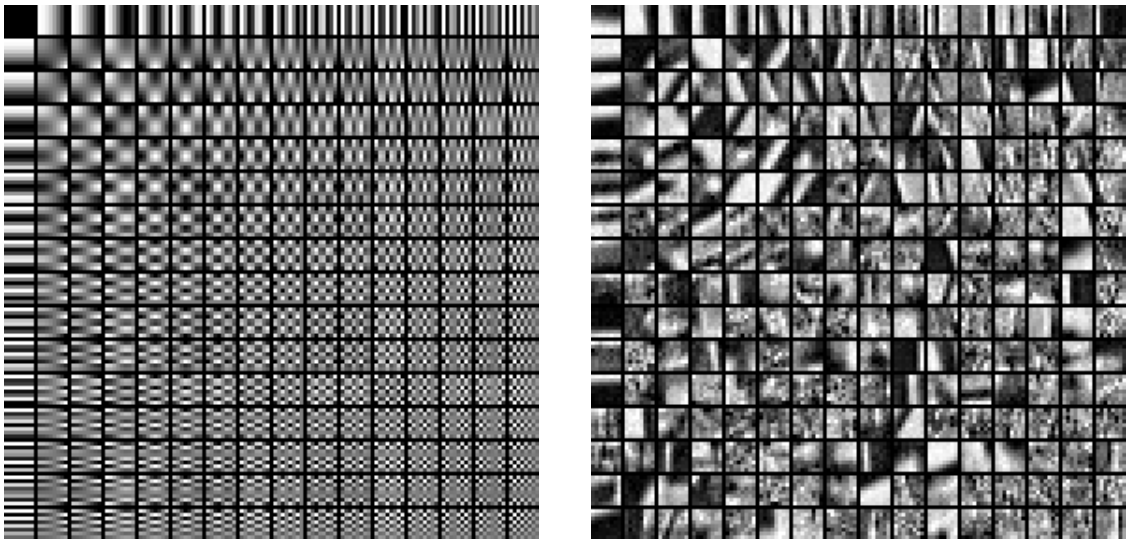
در شکل ۶–۱۹ اتم‌های دیکشنری فوق کامل DCT و نیز اتم‌های دیکشنری آموزش دیده با الگوریتم PAU-DL برای تصویر Boat و SNR = ۲۰ dB آورده شده است[۱]. این اتم‌ها به ترتیبِ واریانس مرتب شده‌اند. با توجه به این شکل، اتم‌های آموزش دیده بهتر از اتم‌های ثابتِ DCT ویژگی‌های تصویر Boat را توضیح می‌دهند. بعنوان مثال، انواع لبه‌های مورّب در میان اتم‌های آموزش دیده وجود دارد حال آن که اتم‌های DCT صرفاً لبه‌های افقی و عمودی را خوب توصیف می‌کنند.

به یاد بیاورید که هر تناوب از آموزش دیکشنری برای نویززدائی از تصاویر شامل دو گام است. ابتدا با شروع از دیکشنری DCT فوق کامل، تخمینی از نسخه‌ی بدون نویزِ بلوک‌هائی از تصویر که بعنوان داده‌ی آموزشی انتخاب شده‌اند با استفاده از این دیکشنری محاسبه می‌شود. این تخمین‌ها با استفاده از نسخه‌ی مقید به خطای

[۱] این شکل‌ها با استفاده از KSVD-Box v13 رسم شده‌اند.



شکل ۶–۱۸: زمان متوسط اجرای سه الگوریتم K-SVD، PAU-DL و RLMC-DL برحسب انحراف معیار نویز.



شکل ۶–۱۹: دیکشنری فوق کامل آموزش دیده با الگوریتم PAU-DL برای تصویر Boat و $\mathrm{SNR} = ۲۰\ \mathrm{dB}$ (سمت راست) و دیکشنری DCT فوق کامل (سمت چپ).

الگوریتم OMP بدست می‌آیند. سپس دیکشنری با استفاده از این بلوک‌ها به‌روز شده و تخمینی دیگر از نسخه‌ی بدون نویز این بلوک‌ها در دیکشنریِ جدید حاصل می‌شود. در نتیجه، هر تناوب از آموزش دیکشنری تخمینی از

جدول ۶-۲: نتایج نویززدائی با دیکشنریِ آموزش دیده توسط سه الگوریتم K-SVD، PAU-DL، RLMC-DL و دیکشنری فوق کامل DCT. در هر سلولِ متناظر با یک تصویر، چهار مقدار برای PSNR خروجی الگوریتم‌های مذکور گزارش شده است. بالا سمت چپ: K-SVD، بالا سمت راست: PAU-DL، پائین سمت چپ: RLMC-DL و پائین سمت راست: دیکشنری DCT فوق کامل.

σ,PSNR	House		Boat		Lena		Barbara		Average	
۲, ۴۲/۱۱	۴۴/۴۸	۴۴/۵۰	۴۳/۶۶	۴۳/۶۷	۴۴/۵۰	۴۴/۵۳	۴۳/۹۶	۴۳/۹۸	۴۴/۱۵	۴۴/۱۵
	۴۴/۴۸	۴۴/۳۹	۴۳/۶۶	۴۳/۶۱	۴۴/۵۰	۴۴/۳۸	۴۳/۹۶	۴۳/۸۴	۴۴/۱۵	۴۴/۰۶
۵, ۳۴/۱۶	۳۹/۴۴	۳۹/۴۸	۳۷/۴۱	۳۷/۴۳	۳۸/۸۶	۳۸/۸۹	۳۸/۱۳	۳۸/۱۸	۳۸/۴۶	۳۸/۵۰
	۳۹/۴۳	۳۹/۰۶	۳۷/۴۰	۳۷/۳۵	۳۸/۸۴	۳۸/۷۲	۳۸/۱۳	۳۷/۸۱	۳۸/۴۵	۳۸/۲۴
۱۰, ۲۸/۱۱	۳۶/۰۸	۳۶/۱۴	۳۳/۳۲	۳۳/۳۷	۳۵/۰۲	۳۵/۱۰	۳۴/۱۰	۳۴/۱۵	۳۴/۶۳	۳۴/۶۹
	۳۶/۰۷	۳۵/۴۱	۳۳/۳۲	۳۲/۹۳	۳۵/۰۳	۳۴/۵۵	۳۴/۱۱	۳۳/۵۲	۳۴/۶۳	۳۴/۱۰
۱۵, ۲۴/۶۲	۳۴/۴۴	۳۴/۵۱	۳۱/۰۴	۳۱/۰۸	۳۲/۸۵	۳۲/۹۱	۳۱/۸۳	۳۱/۸۶	۳۲/۵۴	۳۲/۵۹
	۳۴/۴۶	۳۳/۴۸	۳۱/۰۵	۳۰/۵۷	۳۲/۸۵	۳۲/۲۲	۳۱/۸۳	۳۱/۱۷	۳۲/۵۵	۳۱/۸۶
۲۰, ۲۲/۱۱	۳۳/۲۷	۳۳/۳۱	۲۹/۵۴	۲۹/۵۷	۳۰/۲۶	۳۰/۲۸	۳۰/۱۷	۳۰/۱۹	۳۰/۸۱	۳۰/۸۴
	۳۳/۲۷	۳۲/۱۴	۲۹/۵۴	۲۸/۹۱	۳۰/۲۵	۳۰/۵۵	۳۰/۱۶	۲۹/۵۲	۳۰/۸۱	۳۰/۲۸
۲۵, ۲۰/۱۶	۳۲/۱۹	۳۲/۲۱	۲۸/۳۳	۲۸/۳۵	۳۰/۰۰	۳۰/۰۴	۲۹/۰۲	۲۹/۰۴	۲۹/۸۹	۲۹/۹۱
	۳۲/۱۳	۳۱/۰۳	۲۸/۳۳	۲۷/۷۱	۳۰/۰۰	۲۹/۳۲	۲۹/۰۰	۲۸/۳۳	۲۹/۸۷	۲۹/۱۰
۵۰, ۱۴/۱۵	۲۸/۰۷	۲۸/۰۸	۲۴/۸۱	۲۴/۸۵	۲۶/۱۴	۲۶/۱۶	۲۵/۳۳	۲۵/۳۴	۲۶/۰۹	۲۶/۱۱
	۲۸/۰۷	۲۷/۵۲	۲۴/۸۰	۲۴/۳۹	۲۶/۱۴	۲۵/۶۳	۲۵/۳۳	۲۴/۹۳	۲۶/۰۹	۲۵/۶۲
۱۰۰, ۸/۱۲	۲۳/۶۸	۲۳/۶۹	۲۱/۸۰	۲۱/۸۲	۲۲/۵۷	۲۲/۶۰	۲۱/۹۰	۲۱/۹۱	۲۲/۴۹	۲۲/۵۱
	۲۳/۶۹	۲۳/۷۳	۲۱/۸۱	۲۱/۸۰	۲۲/۵۸	۲۲/۵۳	۲۱/۹۰	۲۱/۸۴	۲۲/۵۰	۲۲/۴۸

بلوک‌های بدون نویز بدست می‌دهد. مرجع [۲۵] فقط از دیکشنری نهائی برای بدست آوردن تخمین بلوک‌های بدون نویز استفاده کرده است؛ امّا به نظر می‌رسد که استفاده از همه‌ی تخمین‌های یک بلوک منجر به نتایج بهتری شود. بعبارت دقیق‌تر، بلوکِ نوعیِ $\mathbf{y}$ را در نظر بگیرید. تخمین نسخه‌ی بدون نویز این بلوک در تناوب k اُم از آموزش دیکشنری عبارت است از:

$$\hat{\mathbf{y}}^{(k)} = \mathbf{D}^{(k)}\mathbf{x}^{(k)}, \tag{۴۹-۶}$$

که $\mathbf{D}^{(k)}$ دیکشنری تناوب k اُم و $\mathbf{x}^{(k)}$ نمایش تُنُک بلوک $\mathbf{y}$ در این دیکشنری است. حال برای بدست آوردن تخمین نهائی بلوک بدون نویز $\mathbf{y}$، از متوسط‌گیری وزن‌دار همه‌ی تخمین‌ها بصورت زیر استفاده می‌کنیم:

$$\hat{\mathbf{y}} = \frac{\sum_k \lambda_k \hat{\mathbf{y}}^{(k)}}{\sum_k \lambda_k}, \tag{۵۰-۶}$$

که λ_k وزنی است که میزان مشارکت $\hat{\mathbf{y}}^{(k)}$ را در ساخت تخمین نهائی نشان می‌دهد. به این ترتیب، انتظار داریم که با انجام این متوسط‌گیری، نویز باقی‌مانده در هر تخمین تضعیف شود.

برای مشاهده‌ی میزان تأثیر این متوسط‌گیری، آزمایش نویززدائی را به کمک الگوریتم PAU-DL و روی دو تصویر Boat و House، یک‌بار با استفاده از ایده‌ی متوسط‌گیری و بار دیگر به روش معمول انجام می‌دهیم. در این آزمایش از همه‌ی بلوک‌های موجود در هر یک از این تصاویر برای آموزش دیکشنری استفاده می‌کنیم. بعنوان معیار مقایسه، از میزان بهبود SNR در استفاده از متوسط‌گیری نسبت به روش معمول استفاده می‌کنیم. مشابه قبل، هر آزمایش را ۵ بار تکرار کرده و روی نتایج میانگین‌گیری انجام می‌دهیم. برای این آزمایش، همه‌ی وزن‌ها را برابر با ۱ قرار داده‌ایم.

نتایج نهائی در جدول ۶-۳ گزارش شده است. این نتایج قابلیت نسبتاً بالای روش متوسط‌گیری را در تضعیف نویز نشان می‌دهد[۱].

جدول ۶-۳: میزان بهبود SNR در استفاده از ایده‌ی متوسط‌گیری با استفاده از الگوریتم PAU-DL و برای دو تصویر Boat و House.

σ,PSNR	۳۰, ۱۸٫۵۸	۴۰, ۱۶٫۰۸	۵۰, ۱۴٫۱۵	۶۰, ۱۲٫۵۷	۷۰, ۱۱٫۲۴	۸۰, ۱۰٫۰۷	۹۰, ۹٫۰۵	۱۰۰, ۸٫۱۲
Boat	۰٫۱۸	۰٫۳۳	۰٫۱۹	۰٫۳۴	۰٫۳۷	۰٫۳۳	۰٫۳۹	۰٫۴۸
House	۰	۰	۰٫۲۲	۰٫۲۶	۰٫۴۹	۰٫۴۹	۰٫۴۲	۰٫۶۴
Average	۰٫۰۹	۰٫۱۷	۰٫۲۱	۰٫۳۰	۰٫۴۳	۰٫۴۱	۰٫۴۱	۰٫۵۶

۶-۸-۴ نتایج شبیه‌سازی الگوریتم SSF

در این قسمت، عملکرد دو الگوریتم SSF و K-subspace را با انجام یک آزمایش روی داده‌های مصنوعی بررسی می‌کنیم. برای این منظور، تعداد پنج زیرفضای با بُعد ۵ از $\mathbb{R}^{۲۰}$ بصورت تصادفی تولید می‌کنیم. بعبارت دیگر، ماتریس‌های $\mathbf{D}_r \in \mathbb{R}^{۲۰\times ۵},\ r = ۱,\ldots,۵$ را با انتخاب درایه‌هایشان از توزیع گوسی با میانگین صفر و واریانس یک تشکیل داده و سپس ستون‌های آن‌ها را نُرمالیزه می‌کنیم. در ادامه، برای هر زیرفضا تعداد ۵۰ داده تولید می‌کنیم. داده‌های هر زیرفضا بصورت $\mathbf{Y}_r = \mathbf{D}_r\mathbf{X}_r$ و با انتخاب درایه‌های ماتریس $\mathbf{X}_r$ از توزیع گوسی با میانگین صفر و

[۱] توجه کنید که الگوریتم‌های مختلف برای نویززدائی با اختلافی در حدود ۰٫۲ یا ۰٫۳ دسی‌بل با هم رقابت می‌کنند؛ لذا این نتایج به نوبه‌ی خود امیدوار کننده است.

جدول ۶-۴: درصد خطای خوشه‌بندی برای دو الگوریتم K-subspace و SSF.

overlap	۰	۱	۲	۳	۴
K-subspace	۵/۲	۶/۴	۹	۱۱/۱	۱۴/۳
SSF	۰	۲	۳/۷۸	۵/۱۳	۸/۴

واریانس واحد، ساخته می‌شوند. پنج حالت مختلف برای زیرفضاها در نظر می‌گیریم. در حالت اول، زیرفضاها کاملاً مستقل از هم بوده و هیچ اتم مشترکی ندارند. در حالت‌های دوم تا پنجم، هر زیرفضا نسبت به زیرفضای قبلی خود به ترتیب ۳،۲،۱ و ۴ اتم مشترک دارد. توجه کنید که حالت‌هائی که زیرفضاها با هم هم‌پوشانی[۱] دارند به مدل UoS نزدیک‌تر است.

برای ارزیابی عملکرد دو الگوریتم SSF و K-subspace، ماتریس $\mathbf{Y} = [\mathbf{Y}_1, \ldots, \mathbf{Y}_5]$ را به این الگوریتم‌ها داده، سپس درصد خطا در خوشه‌بندی داده‌ها[۲] را برای هر الگوریتم محاسبه می‌کنیم. همانطور که پیش‌تر گفتیم، در الگوریتم K-subspace باید تعداد زیرفضاها معلوم باشد ولی در الگوریتم SSF اینطور نیست. برای الگوریتم SSF، مقدار اولیه‌ی σ برابر با ۱۰۰ و پارامتر ζ برابر با ۰/۱ انتخاب شده است. برای کاهش اثرات تصادفی، هر آزمایش (متناظر با یک مقدار هم‌پوشانی) را ۳۰ بار تکرار کرده، روی نتایج میانگین‌گیری می‌کنیم. نتایج این آزمایش در جدول ۶-۴ آورده شده است. در این آزمایش، زمان متوسط اجرای الگوریتم K-subspace برابر با ۰/۲۹ ثانیه و الگوریتم SSF برابر با ۰/۲۳ ثانیه بوده است. از روی این نتایج مشخص است که الگوریتم SSF درصد خطای کم‌تری نسبت به الگوریتم K-subspace دارد.

۶-۹ جمع‌بندی

در این فصل تعدادی الگوریتم جدید برای آموزش دیکشنری معرفی کردیم. اولین الگوریتم، DL1 بود که در واقع هدف از پیشنهاد آن پُر کردن فاصله‌ی موجود بین دو الگوریتم K-SVD و MOD است. در K-SVD اتم‌ها «یکی یکی» به همراه درایه‌های غیرصفر نمایه‌ی خود به‌روز می‌شوند ولی در MOD همه‌ی اتم‌ها «یک‌جا» به‌روز می‌شوند و در عین حال، درایه‌های غیرصفر نمایه‌ی آن‌ها ثابت می‌ماند. DL1 تلفیقی از این دو الگوریتم است که در آن مشابه MOD اتم‌ها یک‌جا به‌روز می‌شوند و مشابه K-SVD درایه‌های غیرصفر نمایه‌ی هر اتم نیز به‌روز

[۱] Overlap

[۲] Miss-classification error

می‌شود. سپس، برای کاهش حجم محاسباتِ DL1، نسخه‌ی سریع آن، یعنی الگوریتم DL2، را معرفی کردیم. در آزمایش‌های انتهای این فصل نیز دیدیم که عملکرد این دو الگوریتم تقریباً مشابه هم بوده ولی زمان اجرای DL2 چند برابر کمتر از DL1 است.

در ادامه، ایده‌ی به‌روز کردن موازی اتم‌ها و الگوریتم PAU-DL را مطرح کردیم. هدف از معرفی PAU-DL، غلبه بر حجم محاسبات زیاد K-SVD و در عین حال، رسیدن به کارآئی در حد آن و حتّی بالاتر است. روش مورد استفاده در PAU-DL مشابه [۵۲]، استفاده از بهینه‌سازی نوبتی به جای SVD است؛ با این تفاوت که در هر تناوب، همه‌ی اتم‌ها بصورت موازی با هم به‌روز می‌شوند. در شبیه‌سازی‌ها هم دیدیم که عملکرد PAU-DL بهتر از K-SVD بوده و از نظر زمان اجرا چندین برابر سریع‌تر از آن است. در نتیجه، در کاربردهای مختلف می‌توان از PAU-DL بعنوان یک الگوریتم سریع و جایگزینی مناسب برای K-SVD استفاده کرد.

الگوریتم بعدی، RLMC-DL بود که ساختار آن از منظر جواب فُرم بسته‌ی نهائیِ دیکشنریِ به‌روز شده در گام دوم، خیلی شبیه MOD است. این الگوریتم در واقع با یک تیر سه نشان می‌زند! به این ترتیب که با رگولاریزه کردن مسأله‌ی به‌روز کردن دیکشنری، هم از بروز جواب‌های نامطلوب به دلیل بدحالت بودن ماتریسی که معکوس می‌شود جلوگیری می‌کند، هم باعث کاهش همبستگی متقابل اتم‌ها در دیکشنری نهائی می‌شود و هم مشابه MOD جواب فُرم بسته (هر چند تقریبی) دارد. در شبیه‌سازی‌ها نیز عملکرد بهتر RLMC-DL را نسبت به MOD و بخصوص در آزمایش روی سیگنال AR(1) دیدیم. هم‌چنین، این الگوریتم در آزمایش‌های انجام گرفته در این فصل کمترین زمان اجرا را نسبت به سایر الگوریتم‌ها داشت.

الگوریتم OS-DL را در ادامه معرفی کردیم. ساختار کلّی این الگوریتم، همانطور که گفته شد، با تمام الگوریتم‌های فعلیِ آموزش دیکشنری متفاوت است. مشابه K-SVD و PAU-DL، الگوریتم OS-DL نیز مبتنی بر به‌روز کردن یکی یکیِ اتم‌ها است؛ با یک تفاوت که در این الگوریتم برخلاف دو الگوریتم مذکور، «تمام» درایه‌های نمایه‌ی هر اتم اجازه‌ی تغییر دارند. برای جلوگیری از پُر شدن نمایه‌ها، یک قید نُرم یک هم اضافه کردیم. در OS-DL، گام اول آموزش دیکشنری به طور صریح انجام نمی‌شود ولی در خلال به‌روز کردن اتم‌ها و نمایه‌ها، درایه‌های هر ستون از ماتریس ضرایب نیز به‌روز می‌شود. الگوریتم OSP-DL همان OS-DL است که در آن مشابه ایده‌ی الگوریتم PAU-DL، اتم‌ها موازیِ با هم به‌روز می‌شوند. در آزمایش‌ها عملکرد رضایت‌بخش این دو الگوریتم را نسبت به الگوریتم‌های دیگر دیدیم.

برای افزایش سرعت همگرائی OS-DL، الگوریتم DL3 را پیشنهاد کردیم. در این الگوریتم، گام نخست آموزش دیکشنری نیز با استفاده از نُرم ℓ_1 انجام می‌شود. اگرچه زمان اجرای DL3 به دلیل انجام این گام افزایش می‌یابد، با این حال امّا سرعت همگرائی آن، همانطور که در آزمایش‌ها دیدیم، نسبت به دو الگوریتم K-SVD و PAU-DL، که نسبت به بقیه عملکرد بهتری دارند، بسیار بیش‌تر است. برای افزایش سرعت اجرای DL3 باید از الگوریتم‌های سریعی که اخیراً معرفی شده‌اند استفاده کرد. ما این موضوع را به کارهای آینده موکول می‌کنیم.

در انتها، الگوریتم SSF را با الهام از بحث نمایش تُنُک و آموزش دیکشنری معرفی کردیم. در این الگوریتم، هدف پیدا کردن یا برازش تعدادی زیرفضا با ابعاد معلوم به یک دسته داده‌ی آموزشی است. ایده‌ی این الگوریتم به این ترتیب است که به طور متوالی تعدادی زیرفضا به داده‌ها برازش کرده و هر بار داده‌های موجود در زیرفضای پیدا شده را از مجموعه‌ی داده‌های آموزشی حذف می‌کند. برای پیاده‌سازی SSF از بهینه‌سازی نوبتی و ایده‌ای مشابه الگوریتم‌های IRLS استفاده کردیم. تفاوت اساسی این الگوریتم با الگوریتم K-subspace در این است که در SSF بر خلاف K-subspace نیازی به دانستن تعداد واقعی زیرفضاها نیست. در آزمایش روی داده‌های مصنوعی دیدیم که الگوریتم SSF عملکرد بهتری نسبت به الگوریتم K-subspace دارد.

فصل ۷

نتیجه‌گیری و پیشنهادات

نمایش تُنُک سیگنال‌ها طی دهه‌ی اخیر کانون توجه تحقیقات زیادی در سراسر جهان بوده است. کارآئی این زمینه از پردازش سیگنال در کاربردهای گوناگونی از جمله بهبود و فشرده‌سازی تصاویر، تشخیص الگو، تصویربرداری پزشکی، جداسازی کور منابع و ... مورد بررسی قرار گرفته و حکایت از توانائی بالای آن در این کاربردها دارد. هدف از نمایش تُنُک، غلبه بر کمبودها و ناتوانائی‌های تبدیل‌های کلاسیک در پردازش و توصیف مناسب سیگنال‌ها است. برای این منظور، به جای استفاده از یک تبدیل کامل، از یک تبدیل فوق کامل، به امید رسیدن به یک نمایش ساده‌تر استفاده می‌شود. این کار معادل بسط دادن سیگنال روی یک مجموعه شامل تعداد زیادی (معمولاً چندین برابر طول سیگنال) بُردار پایه است که «اتم» خوانده می‌شوند. مجموعه‌ی متشکل از این اتم‌ها یک «دیکشنری» نامیده می‌شود. این اتم‌ها باید نمایانگر ویژگی‌های بارز سیگنال بوده و به تعبیری هم‌خانواده‌ی با آن باشند. در نتیجه، ابتدا باید یک دیکشنری مناسب برای یک کلاس مشخص از سیگنال‌ها پیدا کنیم. این موضوع منجر به پیدایش شاخه‌ای از بحث نمایش تُنُک، موسوم به «آموزش دیکشنری» شده است.

در این پایان‌نامه و در راستای پژوهش‌های صورت گرفته در طول چند سال گذشته، پردازش تُنُک سیگنال‌ها را مورد بررسی قرار دادیم. در فصل نخست، مباحث تئوری نمایش تُنُک را شامل قضایای یکتائی و پایداری جواب تُنُک به اختصار بررسی کردیم. ابتدا با بیان قضیه‌ی یکتائی که در سال‌های اخیر اثبات شده است، دیدیم که نمایش به اندازه‌ی کافی تُنُکِ یک سیگنال در یک دیکشنری فوق کامل، یکتا است. سپس پایداری مسائل بازیابی نمایش تُنُک را بررسی کردیم. در ادامه، تعدادی از الگوریتم‌های موجود برای بازیابی تُنُک‌ترین نمایش یک سیگنال را

مرور کردیم. گفتیم که این الگوریتم‌ها در حالت کلی به دو دسته‌ی الگوریتم‌های حریص و الگوریتم‌های مبتنی بر حل یک مسأله‌ی بهینه‌سازی تقسیم می‌شوند. در دسته‌ی اول، به امید رسیدن به یک نمایش تُنُک، مرحله به مرحله سیگنال تحت بررسی با تعدادی از اتم‌های دیکشنری تقریب زده می‌شود. این الگوریتم‌ها خود به دو دسته تقسیم می‌شوند. در دسته‌ی اول، در هر مرحله تنها یک اتم که بیشترین شباهت را به باقی‌مانده‌ی نمایش سیگنال دارد انتخاب می‌شود، مانند دو الگوریتم MP و OMP. در دسته‌ی دوم امّا در هر مرحله چندین اتم به عنوان کاندید حضور در نمایش سیگنال انتخاب می‌شوند، مانند دو الگوریتم CoSaMP و StOMP. به طور کلی، سرعت الگوریتم‌های حریص از دسته‌ی دوم الگوریتم‌ها بالاتر بوده ولی این سرعت بالا به قیمت دقت کمِ جواب نهائی است. در انتهای این فصل، حسگری فشرده را، که در واقع کاربردی از پردازش تُنُک سیگنال‌ها در بحث نمونه‌برداری و ضبط داده‌ها است، معرفی کردیم.

در فصل سوم، آموزش دیکشنری را مطرح کردیم. ابتدا مروری مختصر داشتیم بر انواع دیکشنری‌ها یا تبدیل‌های مختلفی که برای نمایش سیگنال‌ها وجود دارد. گفتیم که مشکل دیکشنری‌های ثابت (کامل یا فوق کامل)، مانند دیکشنری فوریه یا دیکشنری DCT، این است که با وجود محاسبات سریع، توانائی وفق کردن به محتویات سیگنال‌های تحت بررسی را ندارند. بعنوان یک دیکشنری وابسته به سیگنال یا به عبارتی، یک دیکشنری وفقی، تبدیل KLT معرفی شد که در واقع یک زیرفضا به داده‌ها برازش می‌کند. این تبدیل که به PCA نیز معروف است، جزء تبدیل‌های کامل بوده و همانطور که گفته شد، توانائی نمایش مناسب ساختارهای پیچیده‌ترِ سیگنال‌ها را ندارد. مثلاً اگر داده‌ها در چندین زیرفضا قرار داشته باشند، این تبدیل تنها «یک» زیرفضا را به داده‌ها برازش می‌کند. اگرچه تبدیل PCA تعمیم‌یافته (GPCA) برای رفع این مشکل پیشنهاد شد، امّا همانطور که توضیح داده شد، مدلی که این تبدیل برای سیگنال‌ها فرض می‌کند عموماً در کاربردهای واقعیِ پردازش تُنُک برقرار نیست. برای نمایش بهتر سیگنال‌ها، بحث آموزش دیکشنری طی چند سال گذشته مطرح شده است که در آن با تعمیم الگوریتم K-means، اتم‌های دیکشنری طوری بهینه می‌شوند که تا حد امکان نمایش تُنُکی برای سیگنال‌ها ارائه دهند. از این دست الگوریتم‌ها تعدادی را به اختصار در این فصل مرور کردیم.

بعنوان کاربردی از پردازش تُنُک، موضوع نویززدائی از تصاویر را در فصل چهارم بررسی کردیم. بطور خلاصه، در این رهیافت برای نویززدائی از یک تصویر، یک دیکشنری با استفاده از بلوک‌های خودِ تصویر نویزی آموزش می‌بینند. حین آموزش دیکشنری، چون در گام بدست آوردن نمایش تُنُک، از نسخه‌ی مقیّد به خطای

الگوریتم OMP استفاده می‌شود، بلوک‌ها در تکرارِ آخرِ آموزش دیکشنری در واقع نویززدائی هم می‌شوند. در انتها، برای بدست آوردن تخمینی از تصویر بدون نویز، بلوک‌های حاصل با انجام یک متوسط‌گیری کنار هم قرار داده می‌شوند. تا جائیکه نگارنده اطّلاع دارد، طی چند سال اخیر کارهای کمی روی این کاربرد از نمایش تُنُک انجام شده است. دلیل این امر احتمالاً حجم محاسبات بالای الگوریتم K-SVD برای آموزش دیکشنری است. به همین دلیل، چند سالی است که نگاه‌ها متوجهِ الگوریتم‌های آنلاین برای آموزش دیکشنری شده است. این الگوریتم‌ها نسبت به الگوریتم‌های مبتنی بر پردازش یک‌جای داده‌ها (یا بطور خلاصه، الگوریتم‌های گروهی)، توانائی کارکردن با حجم وسیعی از داده‌ها را دارند. این موضوع بخصوص برای پردازش تصاویرِ با رزولوشن بالا اهمیت پیدا می‌کند. با این وجود امّا الگوریتم‌های گروهی این حُسن را (لااقل در ابعاد نه چندان بالا) دارند که با پردازش یک‌جای داده‌ها، از نسخه‌ی گروهیِ الگوریتم OMP استفاده می‌کنند که این خود منجر به صرفه‌جوئی زیاد در حجم محاسباتِ گام نخست آموزش دیکشنری می‌شود؛ حال آنکه، الگوریتم‌های آنلاین به دلیل پردازش تنها یک داده بعد از هر بار به‌روز کردن دیکشنری، از این مزیت برخوردار نیستند.

در فصل پنجم، الگوریتم IRLS تعمیم‌یافته (GIRLS) را معرفی کردیم. در این فصل ابتدا چند تعبیر مختلف از نحوه‌ی کار الگوریتم‌های IRLS بیان کرده و سپس با الهام از این تعبیرها، حالت کلی‌تر این الگوریتم‌ها را معرفی کردیم. در الگوریتم GIRLS، برخلاف الگوریتم‌های IRLS، ماتریس وزن‌ها غیر قطری است. با یک دید شهودی توضیح دادیم که در این وضعیت با انتخاب مناسب درایه‌های این ماتریس، همگرائی و دقت جواب نهائی بالا خواهد رفت. در ادامه، تعبیری آماری از الگوریتم GIRLS بیان کردیم. از این نگاه، الگوریتم GIRLS، برخلاف الگوریتم‌های IRLS، از همبستگی میان اتم‌ها استفاده می‌کند. در انتها با انجام شبیه‌سازی، توانائی بالاتر این الگوریتم را در بازیابی نمایش تُنُک نسبت به الگوریتم‌های IRLS دیدیم. زمانِ اجرای GIRLS امّا بیشتر از IRLS بوده که لازم است در آینده روی آن کار شود.

در فصل ششم، چند الگوریتم جدید برای آموزش دیکشنری معرفی کردیم. این الگوریتم‌ها عبارتند از DL1، DL2، PAU-DL، RLMC-DL، OS-DL و DL3. الگوریتم SSF را نیز در انتها و بعنوان الگوریتمی که چندین زیرفضا به داده‌ها برازش می‌کند، معرفی کردیم. الگوریتم DL1 چیزی بین دو الگوریتم K-SVD و MOD است؛ یعنی اولاً شبیه MOD همه‌ی اتم‌ها را یک‌جا به‌روز می‌کند و ثانیاً، شبیه K-SVD مقدار درایه‌های غیر صفر نمایه‌ی آن‌ها را نیز به‌روز می‌کند. DL2 هم نسخه‌ای تقریبی و سریع‌تر از DL1 است. الگوریتم PAU-DL برای غلبه بر

حجم محاسبات بالای K-SVD پیشنهاد شد و همانطور که در شبیه‌سازی‌ها دیدیم، عملکرد آن به طور متوسط از K-SVD بهتر است. الگوریتم RLMC-DL ساختاری مشابه MOD داشته ولی عملکرد آن در آزمایش‌های مختلف از MOD بهتر است. در این الگوریتم، قیدِ نُرم فروبینیوس ماتریسِ گرام دیکشنری به خطای کلّی اضافه شده و در انتها یک جواب تقریبی ولی به فُرم بسته، مشابه MOD، برای آن بدست آوردیم. الگوریتم OS-DL گام نخستِ آموزش دیکشنری را بطور صریح انجام نمی‌دهد و در عوض، در گام به‌روز کردن دیکشنری، همه‌ی درایه‌های نمایه‌ی اتم‌ها را به‌روز می‌کند. الگوریتم DL3 همان OS-DL بوده با این تفاوت که در آن، گام نخستِ آموزش دیکشنری نیز انجام می‌شود. در شبیه‌سازی‌ها دیدیم که عملکرد متوسط OS-DL از الگوریتم‌های مشابه آن، مانند K-SVD و PAU-DL، بهتر است. هم‌چنین، الگوریتم DL3 بهترین سرعت همگرائی را در بین الگوریتم‌های مختلف دارد امّا با این وجود لازم است در آینده روی کاهش زمان اجرای آن کار شود. الگوریتم SSF، مشابه GPCA، هدفش برازش چندین زیرفضا به داده‌ها است و عملکرد آن وقتی بالاست که ساختار داده‌ها واقعاً متشکل از چند زیرفضا باشد. با انجام شبیه‌سازی‌ها و مقایسه‌ی آن با الگوریتم K-subspace، عملکرد بهتر SSF را دیدیم.

در ادامه، چند جهت برای کارهای آینده بیان می‌کنیم. به تعدادی از این موارد در فصل‌های گذشته اشاره کردیم.

۱. توسعه‌ی الگوریتم‌های آنلاین برای آموزش دیکشنری. این الگوریتم‌ها توانائی کارکردن با تعداد زیادی داده‌ی آموزشی را دارند که این خود در ابعاد خیلی بالا اهمیت زیادی دارد.

۲. توسعه‌ی نسخه‌ی گروهی الگوریتم‌های بدست آوردن نمایش تُنُک. مزیت بزرگی که الگوریتم OMP دارد این است که نسخه‌ی گروهی آن وجود داشته و این منجر به صرفه‌جوئی زیاد در حجم محاسبات می‌شود. ایراد OMP امّا دقت کمِ جواب‌های آن است که این خود منجر به استفاده از الگوریتم‌های مبتنی بر حل مسأله‌ی بهینه‌سازی می‌شود. با توجه به زمان اجرای زیاد این دسته از الگوریتم‌ها، لازم است نسخه‌ی گروهیِ آن‌ها را بدست آوریم. این موضوع بخصوص در کاربردهائی مانند نویززدائی از تصاویر اهمیت می‌یابد. در این کاربردها، نسخه‌ی گروهی الگوریتم OMP تأثیر زیادی روی زمان اجرای کلّی دارد؛ ولی عیب آن این است که دقت OMP پائین بوده و در واقع در انتخاب اتم‌های مناسب دچار اشتباه می‌شود.

۳. توسعه‌ی الگوریتم GIRLS. این الگوریتم همانطور که گفته شد نیاز به بررسی بیشتری دارد.

۴. تلاش در جهت اثبات این که آیا واقعاً یک دیکشنری تُنُک‌کننده برای کاربردهای واقعی، مانند نویززدائی، وجود دارد یا خیر و اگر جواب مثبت است، ارائه‌ی الگوریتمی برای رسیدن به آن. در فصل سوم، قضیه‌ی یکتائی دیکشنری را معرفی کردیم. فرض این قضیه امّا بسیار محدود کننده بوده و در مورد کاربردهای واقعی برقرار نیست. از طرفی، الگوریتم‌های فعلی، یک تعدادِ مشخص اتم فرض می‌کنند که اصلاً معلوم نیست آیا این تعداد بهینه هست یا خیر. بعبارت دیگر، باید بررسی شود که آیا می‌توان چند اتم را که هم از نظر تعداد و هم از نظر ارائه‌ی نمایشی تُنُک برای سیگنال‌ها بهینه هستند پیدا کرد یا خیر.

۵. ارائه‌ی الگوریتمی برای آموزش دیکشنری که در آن اتم‌ها، یا همان ویژگی‌های برجسته، یکی یکی از داده‌های آموزشی استخراج می‌شوند. چنین الگوریتمی دیدی شهودی‌تر نسبت به سایر الگوریتم‌ها داشته و در آن لازم نیست تعداد اتم‌ها از ابتدا معلوم فرض شود.

۶. استفاده از الگوریتم‌های سریعی که در چند سال گذشته معرفی شده است برای گام نخست الگوریتم DL3. هم‌چنین، بکارگیری این ایده که با افزایش مقدار پارامتر λ، زمان اجرای این الگوریتم و الگوریتم OS-DL کم‌تر می‌شود.

مراجع

[1] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[2] M. Aharon, M. Elad, and A. M. Bruckstein, "On the uniqueness of overcomplete dictionaries, and a practical way to retrieve them," *Journal of Linear Algebra and Applications*, vol. 416, pp. 48–67, 2006.

[3] M. Babaie-Zadeh and C. Jutten, "On the stable recovery of the sparsest overcomplete representations in presence of noise," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5396–5400, 2010.

[4] R. G. Baraniuk, "Compressive sensing," *IEEE Signal Proc. Magazine*, vol. 24, no. 4, pp. 118–121, 2007.

[5] R. G. Baraniuk, V. Cevher, M. F. Duarte, and C. Hegde, "Model-based compressive sensing," *IEEE Trans. on Information Theory*, vol. 56, no. 4, pp. 1982–2001, 2010.

[6] R. G. Baraniuk, V. Cevher, and M. B. Wakin, "Low-dimensional models for dimensionality reduction and signal recovery: A geometric perspective," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 959–971, 2010.

[7] A. Blake and A. Zisserman, *Visual Reconstruction*, MIT Press, Cambridge, 1987.

[8] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Applied and Computational Harmonic Analysis*, vol. 27, no. 3, pp. 265–274, 2009.

[9] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Info. Theory*, vol. 52, no. 2, pp. 489–509, 2006.

[10] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Proc. Magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[11] J. V. Candy, *Model-Based Signal Processing*, John Wiley & Sons, 2006.

[12] V. Cevher and A. Krause, "Greedy dictionary selection for sparse representation," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 5, pp. 979–988, 2011.

[13] R. Chartrand and W. Yin, "Iteratively reweighted algorithms for compressive sensing," in *IEEE ICASSP*, 2008.

[14] S. S. Chen, D. D. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Rev.*, vol. 43, pp. 129–159, 2001.

[15] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Trans. on Image Processing*, vol. 16, no. 8, pp. 2080–2095, 2007.

[16] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Trans. on Information Theory*, vol. 55, no. 5, pp. 2230–2249, 2009.

[17] I. Daubechies, M. Defrise, and C. De-Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Comm. Pure Appl. Math.*, vol. 57, no. 11, pp. 1413–1457, 2004.

[18] D. L. Donoho, "Compressed sensing," *IEEE Trans. on Information Theory*, vol. 52, no. 4, pp. 1289–1306, April 2006.

[19] D. L. Donoho and M. Elad, "Optimally sparse representation in general (nonorthogonal) dictionaries via $\ell^{\backslash}$ minimization," *Proc. Nat. Aca. Sci*, vol. 100, no. 5, pp. 2197–2202, 2003.

[20] D. L. Donoho, M. Elad, and V. Temlyakov, "Stable recovery of sparse overcomplete representations in the presence of noise," *IEEE Trans. Info. Theory*, vol. 52, no. 1, pp. 6–18, 2006.

[21] D. L. Donoho and X. Huo, "Uncertainty principles and ideal atomic decomposition," *IEEE Trans. Information Theory*, vol. 47, no. 7, pp. 2845–2862, 2001.

[22] D. L. Donoho and I. M. Johnstone, "Wavelet shrinkage–Asymptopia," *J. Roy. Statist. Soc.*, vol. 57, no. 2, pp. 301–337, 1995.

[23] D. L. Donoho, Y. Tsaig, I. Drori, and J. L. Starck, "Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit," *IEEE Trans. on Information Theory*, vol. 58, no. 2, pp. 1094–1121, 2012.

[24] M. Elad, *Sparse and Redundant Representations*, Springer, 2010.

[25] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. on Image Processing*, vol. 15, no. 12, pp. 3736 – 3745, 2006.

[26] M. Elad, M. A. T. Figueiredo, and Y. Ma, "On the role of sparse and redundant representations in image processing," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 972–982, 2010.

[27] M. Elad, B. Matalon, J. Shtok, and M. Zibulevsky, "A wide-angle view at iterated shrinkage algorithms," in *in Proc. SPIE (Wavelet XII)*, 2007, pp. 26–29.

[28] Y. C. Eldar and M. Mishali, "Robust recovery of signals from a structured union of subspaces," *IEEE Trans. on Information Theory*, vol. 55, no. 11, pp. 5302–5316, 2009.

[29] K. Engan, S. O. Aase, and J. Hakon Husoy, "Method of optimal directions for frame design," in *Proceedings of IEEE ICASSP*, 1999, vol. 5.

[30] F. Faktor, Y. C. Eldar, and M. Elad, "Exploiting statistical dependencies in sparse representations for signal recovery," *IEEE Trans. on Signal Processing*, vol. 60, no. 5, pp. 2286–2303, 2012.

[31] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Springer, 1992.

[32] R. G. Gonzalez and R. E. Woods, *Digital Image Processing*, Prentice Hall, 2007.

[33] B. V. Gowreesunker and A. H. Tewfik, "Learning sparse representation using iterative subspace identification," *IEEE Trans. on Signal Proc.*, vol. 58, no. 6, pp. 3055–3065, 2010.

[34] R. Gribonval and K. Schnass, "Dictionary identification sparse matrix-factorization via ℓ_1 -minimization," *IEEE Trans. Info. Theory*, vol. 56, no. 7, pp. 3523–3539, 2010.

[35] E. T. Hale, W. Yin, and Y. Zhang, "A fixed-point continuation method for ℓ_1 -regularized minimization with applications to compressed sensing," Tech. Rep., Rice University CAAM Tech. Report, July 2007.

[36] J. Ho, M. Yang, J. Lim, K. Lee, and D. Kriegman, "Clustering appearances of objects under varying illumination conditions," in *Computer Vision and Pattern Recognotion*, 2003, vol. 1, pp. 11–18.

[37] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[38] I. T. Jolliffe, *Principal Component Analysis*, Springer, second ed. edition, 2002.

[39] J. Kovacevic and A. Chebira, "Life beyond bases: The advent of frames (Part I)," *IEEE Signal Proc. Magazine*, vol. 24, no. 4, pp. 86–104, 2007.

[40] K. Kreutz-Delgado, J. Murray, B. Rao, K. Engan, T. Lee, and T. Sejnowski, "Dictionary learning algorithms for sparse representation," *Neural Comp.*, vol. 15, no. 2, pp. 349–396, 2003.

[41] K. Labusch, E. Barth, and T. Martinetz, "Robust and fast learning of sparse codes with stochastic gradient descent," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 5, pp. 1048–1060, 2011.

[42] M. S. Lewicki and T. Sejnowski, "Learning overcomplete representations," *Neural Comp.*, vol. 12, no. 2, pp. 337–365, 2000.

[43] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly, "Compressed sensing MRI," *IEEE Signal Proc. Magazine*, vol. 25, no. 2, pp. 72–82, 2008.

[44] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *Journal of Machine Learning Research*, vol. 11, pp. 19–60, 2010.

[45] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. on Signal Proc.*, vol. 41, no. 12, pp. 3397–3415, 1993.

[46] H. Mohimani, M. Babaie-Zadeh, and Ch. Jutten, "A fast approach for overcomplete sparse decomposition based on smoothed ℓ° norm," *IEEE Trans. on Signal Processing*, vol. 57, pp. 289–301, 2009.

[47] D. Needell and J. A. Tropp, "Cosamp: Iterative signal recovery from in-complete and inaccurate samples," *Appl. Comput. Harmon. Anal.*, vol. 26, no. 3, pp. 301–321, 2009.

[48] B. A. Olshausen and D. J. Field, "Natural image statistics and efficient coding," *Network Comp. Neural Syst.*, vol. 7, no. 2, pp. 333–339, 1996.

[49] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: a strategy employed by V1?," *Vision Research*, vol. 37, pp. 3311–3325, 1997.

[50] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in *In Proc. Asilomar Conf. Signal Syst. Comput.*, 1993.

[51] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1045–1057, 2010.

[52] R. Rubinstein, M. Zibulevsky, and M. Elad, "Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit," Tech. Rep., Technion University, 2008.

[53] K. Skretting and K. Engan, "Recursive least squares dictionary learning algorithm," *IEEE Trans. on Signal Processing*, vol. 58, pp. 2121 – 2130, 2010.

[54] I. Tosic and P. Frossard, "Dictionary learning," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 27–38, 2011.

[55] J. A. Tropp, "Greed is good: algorithmic results for sparse approximation," *IEEE Trans. Inform. Theory*, vol. 50, no. 10, pp. 2231–2242, 2004.

[56] J. A. Tropp and S. J. Wright, "Computational methods for sparse solution of linear inverse problems," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 948–958, 2010.

[57] M. Vetterli and J. Kovacevic, *Wavelets and subband coding*, Prentice-Hall, 2007.

[58] R. Vidal, Y. Ma, and S. Sastry, "Generalized principal component analysis (GPCA)," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 12, pp. 1945–1959, 2005.

[59] Y. Wang and W. Yin, "Sparse signal reconstruction via iterative support detection," *SIAM Journal on Imaging Sciences*, vol. 3, no. 3, pp. 462–491, 2010.

[60] D. Wipf and S. Nagarajan, "Iterative re-weighted ℓ_1 and ℓ_2 methods for finding sparse solutions," *Journal of Selected Topics in Signal Processing (Special Issue on Compressive Sensing)*, vol. 4, no. 2, 2010.

[61] M. Yaghoobi, T. Blumensath, and M. Davies, "Regularized dictionary learning for sparse approximation," in *European Signal Processing Conference (EUSIPCO)*, 2008.

[62] M. Yaghoobi, T. Blumensath, and M. E. Davies, "Dictionary learning for sparse approximations with the majorization method," *IEEE Trans. on Signal Processing*, vol. 57, no. 6, pp. 2178 – 2191, 2009.

[63] Y. Zhang and N. Kingsbury, "Fast l0-based sparse signal recovery," in *IEEE Int. Workshop on Machine Learning for Signal Proc. (MLSP)*, 2010, pp. 403–408.

ABSTRACT

Sparse signal processing (SSP), as a powerful tool and an efficient alternative to traditional complete transforms, has become a focus of attention during the last decade. In this approach, we want to approximate a given signal as a linear combination of as few as possible basis signals. Each basis signal is called an atom and their collection is called a dictionary. This problem is in general difficult and belongs to the Np-hard problems; since it requires a combinatorial search. In recent years however, it has been shown both theoretically and experimentally that the sparset possible representation of a signal in an overcomplete dictionary is unique under some conditions and can be found in polynomial time. Consequently, this subject was rapidly used in many applications such as data compression, blind source separation (BSS), image enhancement, medical imaging, pattern recognition, and so on. There are two important problems in SSP. One is to find an appropriate overcomplete dictionary for a given class of signals, i.e. a dictionary that provides sufficient sparse representation for all members of that class. This has led to the development of the dictionary learning algorithms. The second problem is to have an efficient algorithm that recovers the sparset possible representation of a signal. This has also led to the development of different sparse coding algorithms. In this thesis, we first review some of the existing algorithms for sparse coding and dictionary learning. We also consider the image denoising problem using sparse representation. We then propose a new sparse coding algorithm. This algorithm is a generalization of the Iteratively Re-weighted Least Squares (IRLS) algorithms. We continue by proposing some new dictionary learning algorithms. Two of these algorithms are indeed a combination of the ideas of K-Singular Value Decomposition (K-SVD) and Method of Optimal Directions (MOD) algorithms. In order to overcome the high computational burden of K-SVD, we propose an efficient algorithm. The simulations performed on both synthetic and real data show the advantage of the proposed algorithm over K-SVD in both execution time and quality of the results. We then propose three new algorithms which differ structurally from the existing algorithms while performing averagely better. In order to improve the performance of image denoising using sparse representation, we propose a method which is based on averaging the different estimations of the image blocks during the learning process. The simulation results show the promising performance of this method.

KEYWORDS

1. Sparse Representation.
2. Compressed Sensing.
3. Dictionary Learning.
4. Image Denoising.

SHARIF UNIVERSITY OF TECHNOLOGY
ELECTRICAL ENGINEERING DEPARTMENT

M.Sc. THESIS

Title:

Sparse Representation and its Application in Image Denoising

by:

Mostafa Sadeghi

Supervisor:

Dr. Massoud Babaie-Zadeh

September 2012