AMP Tools for Large-Scale Inference

Prof. Philip Schniter

THE OHIO STATE UNIVERSITY

Supported in part by NSF grant CCF-1018368, by NSF grant 1218754, by NSF-I/UCRC grant IIP-0968910, and by DARPA/ONR grant N66001-10-1-4090.

Seminar @ OSU Laboratory for Artificial Intelligence
11/8/2013

Sparse Linear Regression

In sparse linear regression, we want to learn a sparse weight vector $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^N$ that matches the observed data

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{w} \in \mathbb{R}^M$$

where

- $\boldsymbol{A} \in \mathbb{R}^{M \times N}$ is a matrix that may represent collected feature data or a physical measurement process (e.g., a blur kernel in image restoration),
- $\boldsymbol{w}$ represents an additive perturbation or modeling error,
- $N \gg M$ in many cases of interest, in which case $\boldsymbol{A}$ is assumed to be a stable embedding from $\mathcal{X}$ to $\mathbb{R}^M$.

Note: We could easily generalize to complex-valued $\boldsymbol{y}, \boldsymbol{A}, \boldsymbol{x}, \boldsymbol{w}$ if needed.

Minimization of regularized squared loss

- A popular approach to recovering $\boldsymbol{x}$ is via the optimization problem

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \tfrac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda G(\boldsymbol{x})$$

  where $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2$ penalizes residual loss, $G(\boldsymbol{x})$ promotes sparsity (e.g., convex $G(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$ or $\|\boldsymbol{x}\|_q^q$ for $q < 1$), and λ is a trade-off parameter.

- A Bayesian interpretation of the above is that $\hat{\boldsymbol{x}}$ is the MAP estimate of $\boldsymbol{x}$ under the prior pdf $f(\boldsymbol{x}) \propto e^{-\lambda G(\boldsymbol{x})/\nu^w}$ and error $\boldsymbol{w} \sim \mathcal{N}(0, \nu^w)$.

- For now, we focus on the simple case of separable regularizers, i.e., $G(\boldsymbol{x}) = \sum_{j=1}^N g_j(x_j)$, such as $\|\boldsymbol{x}\|_1$ and $\|\boldsymbol{x}\|_q^q$, which corresponds to a statistically independent weight prior, i.e., $f(\boldsymbol{x}) = \prod_{j=1}^N f_j(x_j)$.

Minimization of mean-squared weight error

- In practice, we may instead want the MSE-optimal estimate of $\boldsymbol{x}$:

  $$\hat{\boldsymbol{x}} = \mathsf{E}\{\boldsymbol{x}|\boldsymbol{y}\} = \int \boldsymbol{x}\, f(\boldsymbol{x}|\boldsymbol{y}) d\boldsymbol{x} \quad \text{for posterior pdf} \quad f(\boldsymbol{x}|\boldsymbol{y}) \propto f(\boldsymbol{y}|\boldsymbol{x}) f(\boldsymbol{x})$$

  rather than the solution to a surrogate optimization problem.

- Assuming error $\boldsymbol{w} \sim \mathcal{N}(0, \nu^w)$ and statistically independent weights,

  $$f(\boldsymbol{x}|\boldsymbol{y}) \propto \prod_{i=1}^{N} \mathcal{N}(y_i; \boldsymbol{a}_i^\mathsf{T} \boldsymbol{x}, \nu^w) \prod_{j=1}^{N} f(x_j),$$

  where $\boldsymbol{a}_i^\mathsf{T}$ denotes the i^{th} row of $\boldsymbol{A}$.

- Due to the $\boldsymbol{a}_i^\mathsf{T} \boldsymbol{x}$ coupling term in the posterior $f(\boldsymbol{x} \,|\, \boldsymbol{y})$, the high-dimensional integral does not decouple and thus exact MMSE inference is computationally intractable.

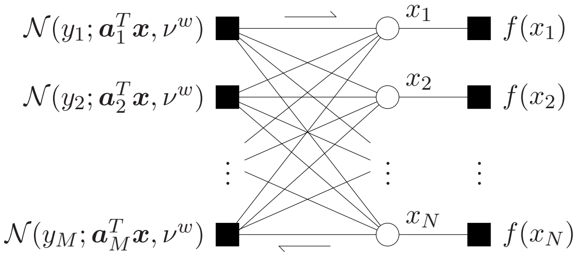
The factor-graph representation

Recall that the previously discussed MAP and MMSE solutions are the maximizer and mean, respectively, of the posterior pdf

$$f(\boldsymbol{x}|\boldsymbol{y}) \propto \prod_{i=1}^{M} \mathcal{N}(y_i; \boldsymbol{a}_i^{\mathsf{T}}\boldsymbol{x}, \nu^w) \prod_{j=1}^{N} f(x_j),$$

which can be visualized using a factor graph:

(White circles are random variables and black boxes are pdf factors.)


$\mathcal{N}(y_1; \boldsymbol{a}_1^T\boldsymbol{x}, \nu^w)$
$\mathcal{N}(y_2; \boldsymbol{a}_2^T\boldsymbol{x}, \nu^w)$
$\mathcal{N}(y_M; \boldsymbol{a}_M^T\boldsymbol{x}, \nu^w)$
x_1
x_2
x_N
$f(x_1)$
$f(x_2)$
$f(x_N)$

Inference via the factor graph: Message passing

- The factor-graph representation leads to two inference algorithms:
  - sum-product algorithm $\rightarrow$ marginal posteriors $\{f(x_j|\boldsymbol{y})\}_{j=1}^N$ $\rightarrow$ MMSE
  - max-sum algorithm $\rightarrow$ MAP

  both of which pass locally computed messages around the graph.

- When the factor-graph contains no loops (i.e., is tree-structured), both methods yield exact estimates, but with loopy graphs (like ours) the inference is *usually* only approximate.

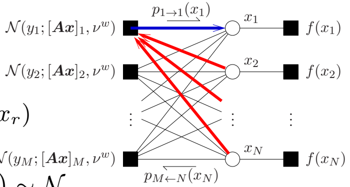
- In any case, the computations needed by the (exact) sum-product and max-sum algorithms are still intractable in the high-dimensional case.

AMP Heuristics (Sum-Product)

1. Message from y_i node to x_j node:

$$p_{i\to j}(x_j) \propto \int_{\{x_r\}_{r\neq j}} \mathcal{N}\big(y_i; \overbrace{\textstyle\sum_r a_{ir} x_r}^{\approx \mathcal{N} \text{ via CLT}}, \psi\big) \prod_{r\neq j} p_{i\leftarrow r}(x_r)$$

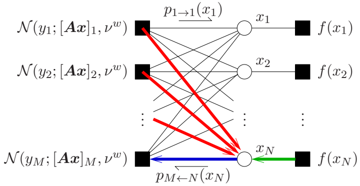
$$\approx \int_{z_i} \mathcal{N}(y_i; z_i, \psi)\, \mathcal{N}\big(z_i; \hat{z}_i(x_j), \nu_i^z(x_j)\big) \sim \mathcal{N}$$


$p_{1\to 1}(x_1)$
$\mathcal{N}(y_1; [\boldsymbol{Ax}]_1, \nu^w)$
$\mathcal{N}(y_2; [\boldsymbol{Ax}]_2, \nu^w)$
$\mathcal{N}(y_M; [\boldsymbol{Ax}]_M, \nu^w)$
x_1
x_2
x_N
$f(x_1)$
$f(x_2)$
$f(x_N)$
$p_{M\leftarrow N}(x_N)$

To compute $\hat{z}_i(x_j), \nu_i^z(x_j)$, the means and variances of $\{p_{i\leftarrow r}\}_{r\neq j}$ suffice, implying Gaussian message passing, like in expectation-propagation.

Remaining problem: we have $2MN$ messages to compute (too many!).

2. Exploiting similarity among the messages $\{p_{i\leftarrow j}\}_{i=1}^M$, AMP employs a Taylor-series approximation of their difference whose error vanishes as $M\to\infty$ for dense $\boldsymbol{A}$ (and similar for $\{p_{i\leftarrow j}\}_{j=1}^N$ as $N\to\infty$).
Finally, need to compute only $\mathcal{O}(M+N)$ messages!


$p_{1\to 1}(x_1)$
$\mathcal{N}(y_1; [\boldsymbol{Ax}]_1, \nu^w)$
$\mathcal{N}(y_2; [\boldsymbol{Ax}]_2, \nu^w)$
$\mathcal{N}(y_M; [\boldsymbol{Ax}]_M, \nu^w)$
x_1
x_2
x_N
$f(x_1)$
$f(x_2)$
$f(x_N)$
$p_{M\leftarrow N}(x_N)$

Approximate message passing (AMP)

When $\boldsymbol{A}$ is large and dense, central-limit-theorem and Taylor-series approximations[1] can be applied to drastically simplify both the sum-product and max-sum algorithms, reducing them to (for avg$\{|a_{ij}|^2\} = \frac{1}{M}$):

```
for $t = 1, 2, 3, \ldots$
    $\hat{\boldsymbol{v}}(t) = \boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{x}}(t) + \frac{N}{M}\frac{\nu^x(t)}{\nu^r(t-1)}\hat{\boldsymbol{v}}(t-1)$    Onsager-corrected residual
    $\hat{\boldsymbol{r}}(t) = \hat{\boldsymbol{x}}(t) + \boldsymbol{A}^{\mathsf{T}}\hat{\boldsymbol{v}}(t)$    back-projection update
    $\nu^r(t) = \nu^w + \frac{N}{M}\nu^x(t)$ or $\frac{1}{M}\|\hat{\boldsymbol{v}}(t)\|_2^2$    error-variance of $\hat{\boldsymbol{r}}(t)$
    $\hat{\boldsymbol{x}}(t+1) = g\big(\hat{\boldsymbol{r}}(t), \nu^r(t)\big)$    nonlinear thresholding step
    $\nu^x(t+1) = \nu^r(t)\,\text{avg}\big\{g'\big(\hat{\boldsymbol{r}}(t), \nu^r(t)\big)\big\}$    error-variance of $\hat{\boldsymbol{x}}(t+1)$
end
```

for $\begin{cases} \text{sum-prod: } g(\hat{r}, v^r) = \mathrm{E}\{X|R=\hat{r}\} \text{ for } R = X + E,\ X \sim f(x),\ E \sim \mathcal{N}(0, \nu^r) \\ \text{max-sum: } g(\hat{r}, v^r) = \mathrm{prox}_{\nu^r f}(\hat{r}) = \arg\min_x\ f(x) + \frac{1}{2\nu^r}(x - \hat{r})^2 \end{cases}$

[1] Donoho, Maleki, Montanari, PNAS 2009 & Rangan, arXiv:1010.5141, 2010.

AMP in perspective

- As described, the inputs to AMP are the weight priors $\{f(x_j)\}_{j=1}^N$, the noise variance ν^w, the choice of sum-product or max-sum, the measurement vector $\boldsymbol{y}$, and the operators $\boldsymbol{A}$ and $\boldsymbol{A}^\mathsf{T}$.
- By choosing appropriate priors $\{f(x_j)\}_{j=1}^M$, one can use AMP to solve many different linear regression problems. For example, to solve the LASSO problem, we'd run max-sum AMP with Laplacian $f(x_j)$.
- The outputs of sum-product AMP are in fact the full marginal posteriors $f(x_j|\boldsymbol{y})$, not only their means, the MMSE estimates $\hat{x}_j$.
- The full marginal posteriors report estimate uncertainty and facilitate tasks such as support detection,[2] tuning,[3] and active learning.[4]

[2] Schniter CISS 2010.

[3] Vila & Schniter SAHD 2011, arXiv:1207.3107.

[4] Schniter CAMSAP 2011.

AMP in perspective (cont.)

- AMP is a so-called first-order algorithm; its computational complexity is dominated by one operation of $\boldsymbol{A}\hat{\boldsymbol{x}}(t)$ and $\boldsymbol{A}^{\mathsf{T}}\hat{\boldsymbol{v}}(t)$ per iteration.
- AMP can directly exploit fast operator implementations of $\boldsymbol{A}$ and $\boldsymbol{A}^{\mathsf{T}}$, such as with Fourier, Wavelet, Hadamard transforms, and even sparse matrices.
- AMP is a form of iterative thresholding that uses an "Onsager" correction term to ensure that

  $\hat{\boldsymbol{r}}(t)$ is an i.i.d-Gaussian corrupted version of the true $\boldsymbol{x}$.

  This concept is key to understanding the how & why of AMP!

AMP in theory

- For large $\boldsymbol{A}$ with entries drawn i.i.d zero-mean sub-Gaussian, a state-evolution[5] characterizes the per-iteration MSE, $\mathrm{E}\{(\hat{X}_j(t)-X_j)^2\}$.

  Morover, when the state-evolution fixed-points are unique, the marginal posterior pdfs $f(x_j|\boldsymbol{y})$ of sum-product AMP converge to the true pdfs, and thus the MMSE estimates $\hat{\boldsymbol{x}}(t)$ become exact.
- For generic $\boldsymbol{A}$, the fixed points[6] of max-sum AMP minimize the optimization objective (i.e., are exact), while those of sum-product AMP minimize a particular variational objective based on independent-Gaussian approximations of KL divergence.
- Note: these analyses study the AMP algorithm itself, not the belief-propagation approximations used to derive AMP.

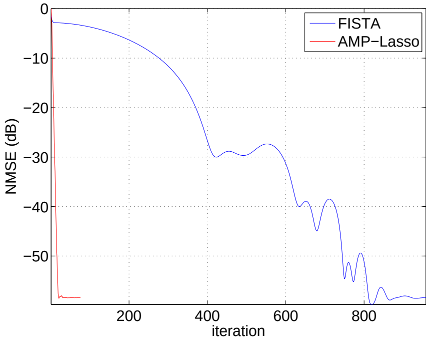
[5] Bayati & Montanari, *arXiv:1001.3448*, 2010

[6] Rangan, Schniter, Riegler, Fletcher, Cevher, *arXiv:1301.6295*, 2013

AMP in practice

- With "well-behaved" $\boldsymbol{A}$, AMP runs much faster than typical sparse linear regression algorithms, e.g., FISTA:

- With "poorly behaved" $\boldsymbol{A}$ (e.g., strongly correlated columns/rows), AMP will diverge unless its iterations are damped.



- An adaptive damping mechanism has been included in the open-source GAMPmatlab toolbox (`http://sourceforge.net/projects/gampmatlab`) that varies the amount of damping so that the objective decreases across iterations.

Choosing weight priors

- As previously described, AMP algorithms can be formulated around different choices of weight prior $f(x_j)$. Note that this prior can vary with the coefficient index j (so we should really be writing $f_{X_j}(x_j)$.)
- In some cases we are forced to work with an established criterion (e.g., LASSO) or we have good prior knowledge of the true $f(x_j)$.
- Then all that remains is to derive the nonlinear thresholding function:

  sum-prod: $g(\hat{r}, v^r) = \mathsf{E}\{X|R=\hat{r}\}$ for $R = X + E$, $X \sim f(x)$, $E \sim \mathcal{N}(0, \nu^r)$
  max-sum: $g(\hat{r}, v^r) = \mathsf{prox}_{\nu^r f}(\hat{r}) = \arg\min_x \; f(x) + \frac{1}{2\nu^r}(x - \hat{r})^2$
- In the case that closed-form expressions do not exist, a scalar Gaussian mixture[7] (GM) approximation can be used to mimic the desired $f(x_j)$ with arbitrarily high accuracy.

[7] Vila and Schniter, arXiv:1207.3107, 2012.

Learning weight priors

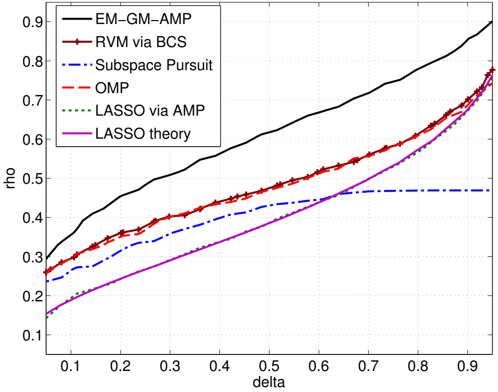
- Often we don't know the weight prior $f(x_j)$ in advance, even though reconstruction MSE would benefit from knowing it.
- Fortunately, in the high dimensional setting, we can learn the weight prior from the noisy compressed measurements $\boldsymbol{y}$.
- For example, we can learn a GM approximation of $f(x_j)$ by using expectation maximization[8] iterations outside AMP, yielding MSE performance virtually indistinguishable from knowing $f(x_j)$ in advance!
- In the high-dimensional limit, the estimates returned by the EM procedure converge to maximum-likelihood estimates.[9]
- In addition, we can simultaneously learn the data-error variance ν^w.

[8] Vila and Schniter, arXiv:1207.3107, 2011.

[9] Kamilov, Rangan, Fletcher, and Unser, arXiv:1207.3859, 2012.

Algorithm comparison 1

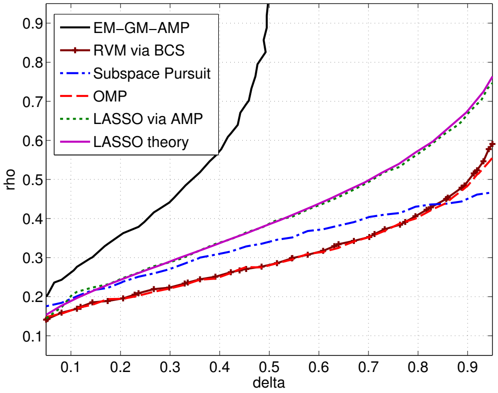
Recall: higher phase-transition-curve = better algorithm.



Here, the non-zero elements of x were drawn independent zero-mean Gaussian. EM-GM-AMP learns and exploits the true weight prior!

Algorithm comparison 1

Recall: higher phase-transition-curve = better algorithm.



Here, the non-zero elements of $\boldsymbol{x}$ were $= 1$.
EM-GM-AMP learns and exploits the true weight prior!

Generalized linear models

- Until now we have assumed linear regression under quadratic loss, i.e., that the observations $\boldsymbol{y}$ are i.i.d-$\mathcal{N}$-corrupted versions of the (hidden) linear transform outputs $\boldsymbol{z} \triangleq \boldsymbol{A}\boldsymbol{x}$:

$$f(\boldsymbol{y}|\boldsymbol{z}) = \prod_{i=1}^{M} f(y_i|z_i) \quad \text{with} \quad f(y_i|z_i) = \mathcal{N}(y_i; z_i, \nu^w)$$

- But there are many applications that need a more general $f(y_i|z_i)$:
  - outliers: $y_i = z_i + w_i$ with super-Gaussian w_m
  - binary classification: $f(y_i|z_i) = [1 + \exp(-y_i z_i)]^{-1}$
  - quantization: $y_i = \text{quant}(z_i)$
  - phase retrieval: $y_i = |z_i|$
  - OFDM comms: $f(y_i|z_i) = s_i z_i + w_i$ with unknown symbol s_i
- Fortunately, the Generalized AMP (GAMP)[10] extension tackles these generalized-linear inference problems.

[10] Rangan, arXiv:1010.5141, 2010.

GAMP in perspective

- GAMP is very similar to AMP but it uses two non-linear thresholding steps: one produces the weight estimate $\hat{\boldsymbol{x}}(t)$ and the other produces the transform estimate $\hat{\boldsymbol{z}}(t)$.
- Max-sum GAMP can be interpreted as a primal-dual algorithm (Arrow-Hurwicz in particular) with adaptively controlled step-sizes.[11]
- Like with AMP, experiments show GAMP running much faster than its peers.
- All AMP theory can be extended to GAMP: the state evolution[12] for large i.i.d sub-Gaussian $\boldsymbol{A}$ and the fixed-point analysis[11] for generic $\boldsymbol{A}$.

[11] Rangan, Schniter, Riegler, Fletcher, Cevher, *arXiv:1301.6295*, 2013

[12] Javanmard and Montanari, arXiv:1211.5164, 2012.

GAMP enables "co-sparse" or "analysis" models

- So far we have been operating under the "synthesis" framework, where $\boldsymbol{x}$ is, say, a sparse (e.g., wavelet) representation of an image $\boldsymbol{s} = \boldsymbol{\Psi x}$, yielding problems like LASSO

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{\Phi\Psi x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1 \text{ and then } \hat{\boldsymbol{s}} = \boldsymbol{\Psi}\hat{\boldsymbol{x}}.$$

  An alternative is the "analysis" framework, e.g., TV regularization

$$\hat{\boldsymbol{s}} = \arg\min_{\boldsymbol{s}} \|\boldsymbol{y} - \boldsymbol{\Phi s}\|_2^2 + \lambda\|\boldsymbol{\Psi}^+\boldsymbol{s}\|_1.$$

- The two are equivalent when the dictionary $\boldsymbol{\Psi}$ is invertible, but not when the dictionary is overcomplete, as is often the case of interest.

- GAMP can be used[13] to solve the analysis problem via the augmentation $\boldsymbol{A} = \left[\begin{smallmatrix}\boldsymbol{\Phi}\\ \boldsymbol{\Psi}^+\end{smallmatrix}\right]$ and appropriate definition of $\{f(y_i|z_i)\}_{i>M}$.

[13] Borgerding, Schniter, Rangan, 2013.

Breaking the independence assumption

- AMP & GAMP were derived under the independence assumptions

$$f(\boldsymbol{x}) = \prod_j f(x_j) \text{ and } f(\boldsymbol{y}|\boldsymbol{z}) = \prod_i f(y_i|z_i).$$

- But in many applications, $\boldsymbol{x}$ or $\boldsymbol{y}|\boldsymbol{z}$ are known to be structured and exploiting this structure can often dramatically aid inference:
  - Persistence-across-time in multi-observation problems
  - Persistence-across-wavelet-scale in natural images
  - Persistence-across-delay in sparse impulse responses
  - Persistence-across-space in change detection
  - Code structure in communications

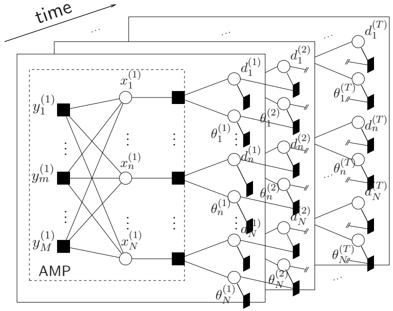
- Such structure can be modeled via structured sparsity (e.g., block-, tree-, field-structured), amplitude correlation, and other methods.

Augmenting the factor graph

As a tangible example, consider recovering a sequence of sparse vectors $\{\boldsymbol{x}^{(l)}\}_{l=1}^{T}$ from the sequence of compressed linear observation vectors

$$\boldsymbol{y}^{(l)} = \boldsymbol{A}\boldsymbol{x}^{(l)} + \boldsymbol{w}^{(l)}, \quad l = 1, \dots, T$$

where $\boldsymbol{x}^{(l)} = \boldsymbol{d}^{(l)} \odot \boldsymbol{\theta}^{(l)}$, with support $\boldsymbol{d}^{(l)} \in \{0,1\}^p$ and amplitudes $\boldsymbol{\theta}^{(l)}$ that both vary slowly over time l.



To tackle such applications, the "turbo AMP" methodology[14] uses sum-product message-passing with AMP approximations in the dense portion of the factor graph.

In this application, turbo-AMP's MSE *nearly matches that of the support-oracle Kalman smoother*.

[14] Schniter, CISS 2010; Ziniel and Schniter, arXiv:1205.4080, 2010.

Learning the structural hyperparameters

- When modeling structure *across* coefficients, one faces the burden of specifing additional hyperparameters.

  For example, on the previous slide, one would need to specify the support transition probabilities $f(d_n^{(l)}|d_n^{(l-1)})$ and the amplitude correlation $\mathsf{E}\{\theta_n^{(l)}\theta_n^{(l-1)}\}$.

- Fortunately, in the high-dimensional regime, these structural hyperparameters can be learned on-the-fly using an EM procedure similar to that discussed earlier.

- An object-oriented implementation[15] of this EM-turbo-AMP methodology is included in the GAMPmatlab toolbox (`http://sourceforge.net/projects/gampmatlab`).

[15] Ziniel, Rangan, and Schniter, SSP 2012.

Generalized-bilinear inference

- Until now we have considered (generalized) linear problems:

  Estimate $\boldsymbol{x}$ given $(\boldsymbol{y}, \boldsymbol{A})$ under likelihood $f(\boldsymbol{y}|\boldsymbol{z})$, where $\boldsymbol{z} = \boldsymbol{A}\boldsymbol{x}$.

- But many important problems are (generalized) bilinear, i.e.,

  Estimate $(\boldsymbol{A}, \boldsymbol{X})$ given $\boldsymbol{Y}$ under likelihood $f(\boldsymbol{Y}|\boldsymbol{Z})$, where $\boldsymbol{Z} = \boldsymbol{A}\boldsymbol{X}$.

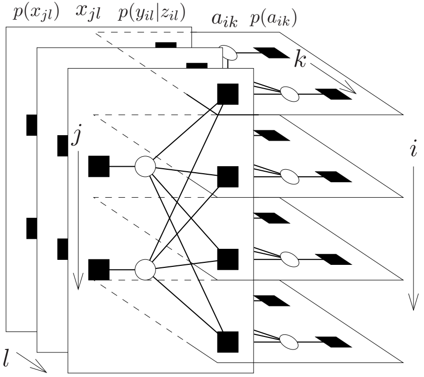
  For example...

  - Matrix completion:
    $\boldsymbol{Z} = \boldsymbol{A}\boldsymbol{X}$ is a low-rank matrix and $f(\boldsymbol{Y}|\boldsymbol{Z})$ hides certain elements.
  - Robust PCA:
    $\boldsymbol{Z} = \boldsymbol{A}\boldsymbol{X}$ is a low-rank matrix and $f(\boldsymbol{Y}|\boldsymbol{Z})$ models outliers.
  - Dictionary learning:
    $\boldsymbol{A}$ is dense, $\boldsymbol{X}$ is sparse, and $f(\boldsymbol{Y}|\boldsymbol{Z})|_{\boldsymbol{Z}=\boldsymbol{A}\boldsymbol{X}}$ models small errors.

Bilinear Generalized AMP (BiG-AMP)

The AMP framework has been applied to the generalized-bilinear factor-graph on the right, yielding the BiG-AMP[16] algorithm.

Furthermore, EM and turbo extensions have been developed for automatic parameter tuning and exploitation of structure *across* the elements of $\boldsymbol{A}$ and $\boldsymbol{X}$.



Experimental results show state-of-the-art performance for BiG-AMP in matrix completion, robust PCA, and dictionary learning applications.

[16] Parker, Schniter and Cevher, ITA 2012, arXiv:1310.2632

Conclusion

- AMP provides a fast and flexible approach to classical sparse linear regression with theoretical guarantees for large i.i.d sub-Gaussian matrices and known fixed-points in general.
- GAMP extends to the generalized linear model, enabling, e.g., logistic regression, phase retrieval, and TV-regularization.
- GAMP can be run inside an expectation-maximization (EM) loop to learn and exploit the true weight prior and data likelihood, since usually these are apriori unknown.
- Turbo-GAMP exploits structure *across* the weights $\{x_j\}$ and the conditional observations $\{y_i|z_i\}$.
- BiG-AMP extends all of the above to generalized bilinear inference problems like matrix completion, robust PCA, and dictionary learning.
- All of the above is implemented in the GAMPmatlab toolbox.