Generalized Approximate Message Passing (GAMP) for Binary Classification and Feature Selection

Phil Schniter and Justin Ziniel

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

ITA (San Diego) — Feb '13

Binary Linear Classification

- Observe m training examples $\{(y_i, \boldsymbol{a}_i)\}_{i=1}^m$, each comprised of a binary label $y_i \in \{-1, 1\}$ and a feature vector $\boldsymbol{a}_i \in \mathbb{R}^n$.

- Assume that data follows a generalized linear model

$$\Pr\{y_i = 1 \,|\, \boldsymbol{a}_i; \boldsymbol{x}_{\text{true}}\} = p_{Y|Z}(1 \,|\, \underbrace{\boldsymbol{a}_i^{\mathsf{T}} \boldsymbol{x}_{\text{true}}}_{\triangleq\, z_{i,\text{true}}})$$

  for some true weight vector $\boldsymbol{x}_{\text{true}} \in \mathbb{R}^p$ and some activation function (or likelihood) $p_{Y|Z}(1, \cdot) : \mathbb{R} \to [0, 1]$.

- Goal 1: estimate $\hat{\boldsymbol{x}}_{\text{train}} \approx \boldsymbol{x}_{\text{true}}$ from training data, so to be able to predict the unknown label y_{test} associated with a test vector $\boldsymbol{a}_{\text{test}}$:

$$\text{compute } \Pr\{y_{\text{test}} = 1 \,|\, \boldsymbol{a}_{\text{test}}; \hat{\boldsymbol{x}}_{\text{train}}\} = p_{Y|Z}(1 \,|\, \boldsymbol{a}_{\text{test}}^{\mathsf{T}} \hat{\boldsymbol{x}}_{\text{train}})$$

Binary Linear Classification & Feature Selection

- Operating regimes:
  - $m \gg n$: Plenty of training examples: feasible to learn $\hat{\boldsymbol{x}}_{\text{train}} \approx \boldsymbol{x}_{\text{true}}$.
  - $m \ll n$: Training-starved: feasible only if $\boldsymbol{x}_{\text{true}}$ is sufficiently sparse!

- The training-starved case motivates...

  Goal 2: Identify salient features (i.e., recover support of $\boldsymbol{x}_{\text{true}}$).

  - Example: From fMRI, learn which parts of the brain are responsible for discriminating two classes of object (e.g., cats vs. houses):

$$\begin{aligned} n = 31398 \quad &\leftrightarrow \quad \text{fMRI voxels} \\ m = 216 \quad &\leftrightarrow \quad \text{2 classes} \times \text{9 examples} \times \text{12 subjects} \end{aligned}$$

  - Can interpret as support recovery in noisy one-bit compressed sensing:

$$\boldsymbol{y} = \operatorname{sgn}(\boldsymbol{A}\boldsymbol{x}_{\text{true}} + \boldsymbol{w}) \text{ with i.i.d noise } \boldsymbol{w}.$$

Bring out the GAMP

Zed: Bring out the Gimp.

Maynard: Gimp's sleeping.

Zed: Well, I guess you're gonna have to go wake him up now, won't you?

—Pulp Fiction, 1994.

We propose a new approach to binary linear classification and feature selection via generalized approximate message passing (GAMP).

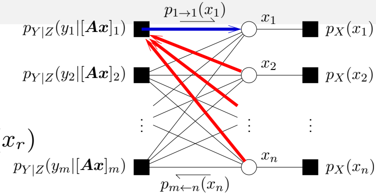
Advantages of GAMP include

- flexibility in choosing likelihood $p_{Y|Z}$ & input prior p_X.
- excellent accuracy & runtime.
- state-evolution governing behavior in some cases.
- can learn & exploit structured sparsity (via turbo extension [S. '10]),
- can tune without cross-validation (via EM extension [Vila & S. '11]),

Generalized Approximate Message Passing (GAMP)

- The evolution of GAMP:
  - The original AMP [Donoho, Maleki, Montanari '09] solves the LASSO problem $\arg\min_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1$ assuming i.i.d sub-Gaussian $\boldsymbol{A}$.
  - The Bayesian AMP [Donoho, Maleki, Montanari '10] extends to MMSE inference in AWGN for any factorizable signal prior $\prod_j p_X(x_j)$.
  - The generalized AMP [Rangan '10] framework extends to MAP or MMSE inference under any factorizable signal prior & likelihood.
- GAMP is a sophisticated form of iterative thresholding, requiring only two applications of $\boldsymbol{A}$ per iteration and few iterations. Very fast!
- Rigorous large-system analyses (under i.i.d sub-Gaussian $\boldsymbol{A}$) have established that GAMP follows a state-evolution trajectory with various nice properties [Rangan '10], [Javanmard, Montanari '12]

GAMP Heuristics (Sum-Product)



1. Message from y_i node to x_j node:

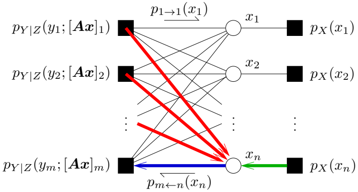
$$p_{i\to j}(x_j) \propto \int_{\{x_r\}_{r\neq j}} p_{Y|Z}\Big(y_i; \overbrace{\textstyle\sum_r a_{ir}x_r}^{\approx\mathcal{N}\text{ via CLT}}\Big) \prod_{r\neq j} p_{i\leftarrow r}(x_r)$$

$$\approx \int_{z_i} p_{Y|Z}(y_i; z_i)\,\mathcal{N}\big(z_i; \hat{z}_i(x_j), \nu_i^z(x_j)\big) \approx \mathcal{N}$$

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, thus Gaussian message passing!

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

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



The GAMP Algorithm

Require: Matrix $\boldsymbol{A}$, sum-prod $\in$ {true,false}, initializations $\hat{\boldsymbol{x}}^0$, $\boldsymbol{\nu}_x^0$

$t = 0$, $\hat{\boldsymbol{s}}^{-1} = \boldsymbol{0}$, $\forall ij : S_{ij} = |A_{ij}|^2$

repeat

$\boldsymbol{\nu}_p^t = \boldsymbol{S}\boldsymbol{\nu}_x^t, \quad \hat{\boldsymbol{p}}^t = \boldsymbol{A}\hat{\boldsymbol{x}}^t - \hat{\boldsymbol{s}}^{t-1}.\boldsymbol{\nu}_p^t$ (gradient step)

if sum-prod **then**

$\forall i : \nu_{z_i}^t = \mathsf{var}(Z|P; \hat{p}_i^t, \nu_{p_i}^t), \quad \hat{z}_i^t = \mathsf{E}(Z|P; \hat{p}_i^t, \nu_{p_i}^t),$

else

$\forall i : \nu_{z_i}^t = \nu_{p_i}^t \mathsf{prox}'_{-\nu_{p_i}^t \log p_{Y|Z}(y_i,.)}(\hat{p}_i^t) \quad \hat{z}_i^t = \mathsf{prox}_{-\nu_{p_i}^t \log p_{Y|Z}(y_i,.)}(\hat{p}_i^t),$

end if

$\boldsymbol{\nu}_s^t = (1 - \boldsymbol{\nu}_z^t./\boldsymbol{\nu}_p^t)./\boldsymbol{\nu}_p^t, \quad \hat{\boldsymbol{s}}^t = (\hat{\boldsymbol{z}}^t - \hat{\boldsymbol{p}}^t)./\boldsymbol{\nu}_p^t$ (dual update)

$\boldsymbol{\nu}_r^t = 1./(\boldsymbol{S}^T\boldsymbol{\nu}_s^t), \quad \hat{\boldsymbol{r}}^t = \hat{\boldsymbol{x}}^t + \boldsymbol{\nu}_r^t.\boldsymbol{A}^T\hat{\boldsymbol{s}}^t$ (gradient step)

if sum-prod **then**

$\forall j : \nu_{x_j}^{t+1} = \mathsf{var}(X|R; \hat{r}_j^t, \nu_{r_j}^t), \quad \hat{x}_j^{t+1} = \mathsf{E}(X|R; \hat{r}_j^t, \nu_{r_j}^t),$

else

$\forall j : \nu_{x_j}^{t+1} = \nu_{r_j}^t \mathsf{prox}'_{-\nu_{r_j}^t \log p_X(.)}(\hat{r}_j^t) \quad \hat{x}_j^{t+1} = \mathsf{prox}_{-\nu_{r_j}^t \log p_X(.)}(\hat{r}_j^t),$

end if

$t \leftarrow t+1$

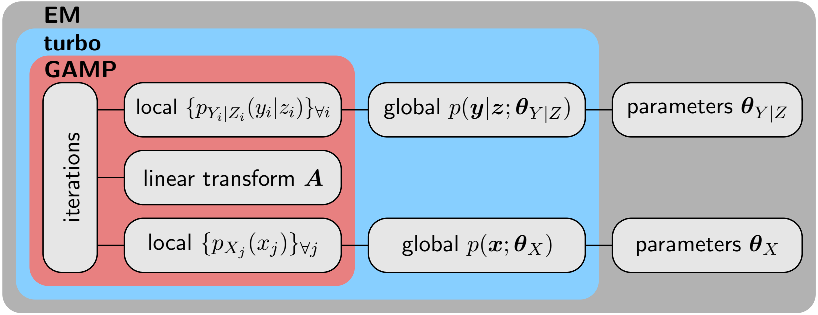
until Terminated

Note connections to Arrow-Hurwicz, primal-dual, ADMM, proximal FB splitting,. . .

Making GAMP Practical: EM & turbo Extensions

- The basic GAMP algorithm requires
  1. separable priors $p(\boldsymbol{y}|\boldsymbol{z}) = \prod_i p_{Y_i|Z_i}(y_i|z_i)$ and $p(\boldsymbol{x}) = \prod_j p_{X_j}(x_j)$
  2. that are perfectly known.

- The EM-turbo-GAMP framework circumvents these limitations by learning possibly non-separable priors:



GAMP for Binary Classification and Feature Selection

- How to use GAMP for binary classification & feature selection? **Mix'n Match** a likelihood $p_{Y|Z}$, prior p_X, and linear transform $\boldsymbol{A}$.
- Our current GAMP implementation includes (among others)

likelihood $p_{Y\|Z}$	sum-prod	max-prod
logit	NI	RF
probit	CF	RF
hinge	CF	RF
robust-*	CF	CF

prior p_X	sum-prod	max-prod
Gaussian	CF	CF
Laplace	CF	CF
Elastic Net	CF	CF
Bernoulli-*	CF	–

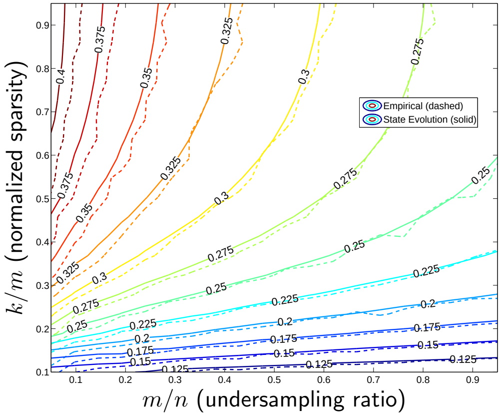
where CF=closed-form, NI=numerical integration, RF=root-finding.

- For linear classification, the rows of GAMP's linear transform $\boldsymbol{A}$ are the feature vectors $\{\boldsymbol{a}_i^\mathsf{T}\}_{\forall i}$. Nonlinear classification is also supported by constructing $[\boldsymbol{A}]_{i,j} = \mathcal{K}(\boldsymbol{a}_i, \boldsymbol{a}_j)$ using an appropriate kernel $\mathcal{K}(\cdot,\cdot)$.

Test Error-Rate via GAMP State Evolution

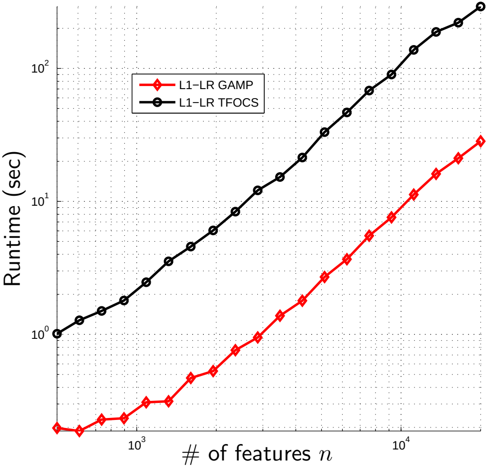
- Recall that, with i.i.d sub-Gaussian $\boldsymbol{A}$ in the large-system limit, GAMP obeys a state evolution that characterizes the accuracy of $\hat{\boldsymbol{x}}_{\text{train}}$ at each iteration t.
- For classification, we can use this SE to predict the test error rate.
- In this example we used $\boldsymbol{A} \sim$ i.i.d $\mathcal{N}(0,1)$, p_X Bernoulli-Gaussian, $p_{Y|Z}$ probit.
- Notice close agreement between SE (solid) and empirical (dashed).

$\Pr\{\text{sgn}(\hat{z}_{\text{test}}) \neq y_{\text{test}}\}$



Runtime Comparison: GAMP vs TFOCS*

- Both algorithms solved the L1-LR problem to tolerance 1×10^{-8}, achieving identical train & testing error rates, but GAMP was an order of magnitude faster.
- Details: $\boldsymbol{A} \in \mathbb{R}^{m \times n} \sim$ i.i.d $\mathcal{N}$, $\boldsymbol{x} \sim k$-sparse BG, $\frac{m}{n} = \frac{1}{3}$ and $\frac{k}{m} = \frac{1}{20}$



*Becker, Candès, Grant, "Templates for convex cone problems with applications to sparse signal recovery," *MPC* 2011.

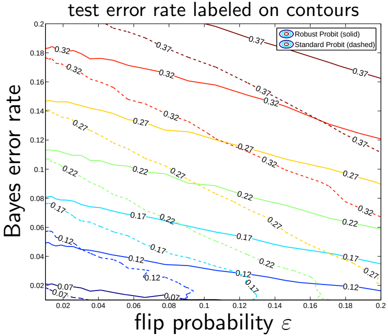
Robust Classification

- Some training sets contain corrupted labels (e.g., randomly flipped).
- For this, GAMP can "robustify" any given likelihood $p_{Y|Z}$ using

$$\tilde{p}_{Y|Z}(y|z) = (1-\varepsilon)p_{Y|Z}(y|z) + \varepsilon\, p_{Y|Z}(1-y|z),$$

where $\varepsilon \in [0,1]$ models the flip probability.

- Here's an example of robust (solid) and non-robust (dashed) GAMP classification performance:



- Details:
  $\boldsymbol{A} \in \mathbb{R}^{300\times 1000} \sim$ non-i.i.d $\mathcal{N}$ with 30-sparse BG $\boldsymbol{x}_{\text{true}}$ and randomly flipped probit $p_{Y|Z}$.

20Newsgroups Example

- 20 different newsgroups were partitioned into two classes (`sci.*`, `comp.*`, `misc.forsale` versus `rec.*`, `talk.*`, `alt.*`, `soc.*`). Goal is to predict the class of a test document from its bag-of-words.

- Data was m =20k examples of n =1.3M features, where feature matrix was 0.0003 sparse. . . far from i.i.d sub-Gaussian!

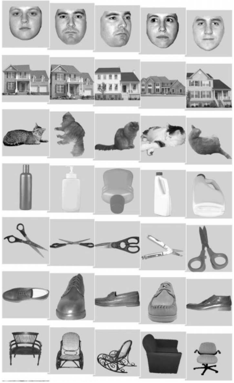
- Test error rate evaluated by 10-fold leave-one-out cross-validation:

algorithm	setup	error rate	runtime
EM-GAMP	sum-prod probit/B-Gauss	3.4%	260 sec
GAMP (cross val)	max-prod logistic/Lap	3.0%	1236 sec
TFOCS (cross val)	logistic/ℓ_1	3.0%	7780 sec

All algorithms terminated based on tol$=1 \times 10^{-4}$.

Haxby Example

- We now return to the problem of learning, from fMRI measurements, which parts of the brain are responsible for discriminating two classes of object.
- Note that the main problem here is feature selection, not classification. The observed classification error rate is used only to judge the validity of the support estimate.
- For this we use the famous Haxby data, with



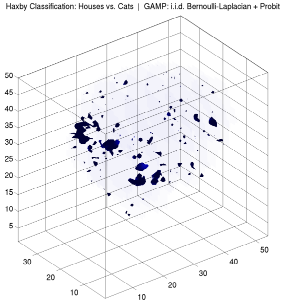
$$n = 31398 \quad \leftrightarrow \quad \text{fMRI voxels}$$
$$m = 216 \quad \leftrightarrow \quad \text{2 classes} \times \text{9 examples} \times \text{12 subjects}$$

Haxby et al., "Distributed and Overlapping Representations of Faces and Objects in Ventral Temporal Cortex" *Science*, 2001.

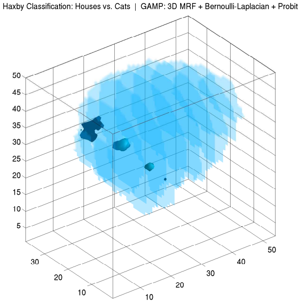
Haxby: Cats vs. Houses

algorithm	setup	error rate	runtime
EM-GAMP	sum-prod probit/B-Gauss	1.4%	9 sec
EM-GAMP	sum-prod probit/B-Laplace	1.9%	13 sec
EM-turbo-GAMP	sum-prod probit/B-Laplace 3D-MRF	1.9%	14 sec

without 3D MRF



with 3D MRF



Conclusions

- We presented preliminary results on the application of GAMP to binary linear classification and feature selection.

- Some nice properties of classification GAMP include
  - flexibility in choice of input and output priors
  - runtime (e.g., $5 - 10\times$ faster than TFOCS)
  - state-evolution can be used to predict test error-rate
  - can handle corrupted labels (via robust prior)
  - can exploit and learn structured sparsity (via turbo extension)
  - can tune without cross-validation (via EM extension), at the expense of a small performance hit.

All these methods are integrated into GAMPmatlab:
`http://sourceforge.net/projects/gampmatlab/`

Thanks!