Linear methods for large data

Dean Foster

(U Penn → Yahoo Labs)
A few weeks ago we had Martinsson present:

“Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions”

- It is my current favorite paper.
- Today, I’ll be applying it to a several problems in ML / statistics
problem  Find a low rank approximation to a $n \times m$ matrix $M$.

solution  Find a $n \times k$ matrix $A$ such that $M \approx AA^\top M$
Basic method

problem  Find a low rank approximation to a $n \times m$ matrix $M$.

solution  Find a $n \times k$ matrix $A$ such that $M \approx AA^T M$

Construction  $A$ is constructed by:

1. create a random $m \times k$ matrix $\Omega$ (iid normals)
2. compute $M\Omega$
3. Compute thin SVD of result: $UDV^T = M\Omega$
4. $A = U$
FAST MATRIX REGRESSIONS
Toy problem: $p \ll n$: 

Solving least squares: (a la Mahoney) 
Generates provably accurate results. Instead of $n^2$ time, it runs in $np$ time. This is fast! (I.e. as fast as reading the data.) 

But we should be unimpressed. 

Alternative fast (but stupid) method: 
Do least squares on a sub-sample of size $n/p$. Runs in time $np$. Same accuracy as the fast methods.
Toy problem: $p \ll n$:

- Solving least squares: (a la Mahoney)
  - Generates provably accurate results.
  - Instead of $np^2$ time, it runs in $np$ time.
  - This is fast! (I.e. as fast as reading the data.)
Toy problem: $p \ll n$:

- Solving least squares: (a la Mahoney)
  - Generates provably accurate results.
  - Instead of $np^2$ time, it runs in $np$ time.
  - This is fast! (i.e. as fast as reading the data.)

- But we should be unimpressed.
Using random methods for regression

Toy problem: $p \ll n$:
- Solving least squares: (a la Mahoney)
  - Generates provably accurate results.
  - Instead of $np^2$ time, it runs in $np$ time.
  - This is fast! (I.e. as fast as reading the data.)
- But we should be unimpressed.
- Alternative fast (but stupid) method:
  - Do least squares on a sub-sample of size $n/p$
  - Runs in time $np$.
  - Same accuracy as the fast methods.
A better fast regression

- Create “sub-sample” \( \hat{\hat{X}} \equiv A^\top X \)
- Estimate
  \[
  \beta = (\hat{\hat{X}}^\top \hat{\hat{X}})^{-1} \hat{\hat{X}}^\top Y
  \]
- Mahoney also subsampled \( Y \) and hence lost accuracy.
New method: Fast and accurate

- As fast as only reading the data (*np* time)
- As accurate as using all the data ([NIPS 2013](#))
New method: Fast and accurate

- As fast as only reading the data (\(np\) time)
- As accurate as using all the data (NIPS 2013)

What about \(p \gg n\)?
New method: Fast and accurate

- As fast as only reading the data ($np$ time)
- As accurate as using all the data (NIPS 2013)

What about $p \gg n$?

- Sub-sample the other side of the $X$ matrix
- Generates a PCAs regression
- Sub-sample columns almost works
- Fast matrix multiply fixes the “almost” (NIPS 2013)
- Aside: yields fast ridge regression also (JMLR 2013)
Outline:

1. Streaming variable selection.
2. Fast CCAs.
3. Fast HMMs.
5. Fast clustering.

All are connected to the fast matrix decomposition.
(1) VIF regression
Basic method: Stream over the features, trying them in order

Even more greedy than stepwise regression (2006)

Provides mFDR protection (2008)

Instead of orthogonalizing each new $X$, only approximately orthogonalize it. (2011)
  - Can be done via sampling
  - Can be done use fast matrix methods

For sub-modular problems, this will generate almost as good an estimator as best subsets. (2013)
Basic method: Stream over the features, trying them in order

Even more greedy than stepwise regression (2006)

Provides mFDR protection (2008)

Instead of orthogonalizing each new $X$, only approximately orthogonalize it. (2011)
  - Can be done via sampling
  - Can be done use fast matrix methods

For sub-modular problems, this will generate almost as good an estimator as best subsets. (2013)
VIF speed comparison

Capacity

Elapsed Running Time

Number of Candidate Variables

- vif-regression
- gps
- stepwise
- lasso
- foba

- vif:100,000
- gps:6,000
- stepwise:900
- lasso:700
- foba:600
(2) CCA for Semi-supervised data
CCA: Usual data table for data mining

\[
\begin{bmatrix}
Y \\
(n \times 1)
\end{bmatrix}
\begin{bmatrix}
X \\
(n \times p)
\end{bmatrix}
\]

with \( p \gg n \)
With unlabeled data

$m$ rows of unlabeled data:

\[
\begin{bmatrix}
Y \\
n \times 1
\end{bmatrix}
\quad \begin{bmatrix}
X \\
(n + m) \times p
\end{bmatrix}
\]
With alternative X’s

$m$ rows of unlabeled data, and two sets of equally useful $X$’s:

\[
\begin{bmatrix}
Y \\
\end{bmatrix}_{n \times 1} \quad \begin{bmatrix}
X \\
(n + m) \times p \\
\end{bmatrix} \quad \begin{bmatrix}
Z \\
(n + m) \times p \\
\end{bmatrix}
\]

With: $m \gg n$
Examples

- **Named entity recognition**
  - Y = person / place
  - X = name itself
  - Z = words before target

- **Modeling words in a sentence**
  - Y = Current word
  - X = previous words
  - Z = future words

- **Sitcom speaker identification:**
  - Y = which character is speaking
  - X = video
  - Z = sound

- We will call these the multi-view setup
We can compute a CCA between $X$ and $Z$ to find a good subspace to use to predict $Y$.

- CCA = canonical correlation analysis
- Finds directions that are most highly correlated
We can compute a CCA between $X$ and $Z$ to find a good subspace to use to predict $Y$.

- CCA = canonical correlation analysis
- Finds directions that are most highly correlated
- Can be solved by doing successive regressions
- So, we can use our fast regression algorithms (2014)
Using a CCA between $X$ and $Z$ to generate features

We can compute a CCA between $X$ and $Z$ to find a good subspace to use to predict $Y$.

- CCA = canonical correlation analysis
- Finds directions that are most highly correlated
- Can be solved by doing successive regressions
- So, we can use our fast regression algorithms (2014)

Results:

- Theory: Using the top few CCA directions is almost as good as the best linear model. (2006)
- We can use this to generate Eigenwords (ICML 2012)
Colors:
- nouns = Blue (dark = NN1, light = NN2)
- verbs = red (dark = VV1, light = VV2)
- adj = green
- unk = yellow
- black = all else

Size = 1/Zipf order, top 50 are solid, rest are open.
(3) HMMs
Hidden Markov Model

HMM with states $h_1$, $h_2$, and $h_3$ which generate observations $x_1$, $x_2$, and $x_3$. 
The $Y$'s are our eigenwords.
The $Y$'s are our eigenwords.

$$\Pr(x_t, \ldots, x_1) = 1^T T \diag(OU^\top y_t) \cdots T \diag(OU^\top y_1) \pi$$
Results

- Sample complexity (2010)
- Empirical results in NLP
  - Named Entity Recognition (CoNLL ’03 shared task)
  - Chunking (CoNLL ’00 shared task)
  - Eigenwords added signal to state of the art systems for both tasks
    - (2011)
- Neural data (2013)
(4) Parsing
We can extend the HMM material to dependency parsing
Same sample complexity (2012)
Raw MST Parser is 91.8% accurate
Adding eigenwords: 2.6% error reduction
eigenwords plus Re-ranking: 7.3% error reduction
Extended to constituent parsing (2014)
(5) Clustering
If you rotate this, you will see there are “pointy” directions
Theorem (with Hsu, Kakade, Liu, Anima, NIPS 2012)

Maximizing $E(\mu^\top X)^4$ will find the natural coordinate system for LDA.
COAUTHORS
## Coauthors

<table>
<thead>
<tr>
<th></th>
<th>U Penn</th>
<th>Elsewhere</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stat faculty</strong></td>
<td>Edward George</td>
<td>Tong Zhang (Rutgers)</td>
</tr>
<tr>
<td></td>
<td>Robert Stine</td>
<td>Sham Kakade (MSR)</td>
</tr>
<tr>
<td><strong>students</strong></td>
<td>Dongyu Lin (ATT)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Jordan Rodu (CMU)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kory Johnson</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yichao Lu</td>
<td></td>
</tr>
<tr>
<td><strong>CS faculty</strong></td>
<td>Lyle Ungar</td>
<td>Michael Collins (Columbia)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Daniel Hsu (Columbia)</td>
</tr>
<tr>
<td><strong>students</strong></td>
<td>Parmaveer Dhillon</td>
<td>Shay Cohen (Columbia)</td>
</tr>
<tr>
<td></td>
<td>Jing Zhou (real world)</td>
<td>Karl Stratos (Columbia)</td>
</tr>
<tr>
<td></td>
<td>U Penn</td>
<td>Elsewhere</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>Stat faculty</td>
<td>Edward George</td>
<td>Tong Zhang (Rutgers)</td>
</tr>
<tr>
<td></td>
<td>Robert Stine</td>
<td>Sham Kakade (MSR)</td>
</tr>
<tr>
<td>students</td>
<td>Dongyu Lin (ATT)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Jordan Rodu (CMU)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kory Johnson</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yichao Lu</td>
<td></td>
</tr>
<tr>
<td>CS faculty</td>
<td>Lyle Ungar</td>
<td>Michael Collins (Columbia)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Daniel Hsu (Columbia)</td>
</tr>
<tr>
<td>students</td>
<td>Parmaveer Dhillon</td>
<td>Shay Cohen (Columbia)</td>
</tr>
<tr>
<td></td>
<td>Jing Zhou (real world)</td>
<td>Karl Stratos (Columbia)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>’14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>’13</td>
</tr>
<tr>
<td></td>
<td>’11 ’12 ’13</td>
<td></td>
</tr>
<tr>
<td></td>
<td>’11 ’12 ’13</td>
<td></td>
</tr>
<tr>
<td>’94 ’00 ’04 ’06</td>
<td>’08 ’10 ’11 ’12 ’13</td>
<td></td>
</tr>
</tbody>
</table>
These new fast matrix methods are easy to program.
Generate statistically useful results.
Room for interesting new probability theory.
- These new fast matrix methods are easy to program.
- Generate statistically useful results.
- Room for interesting new probability theory.

Thanks!
If $n > p^3$, then the algorithm defined by:

- Let $m = \sqrt{n}$
- Pull out a sub-sample of size $m$ from $X$'s and call it $Z$.
- Let $\hat{\beta} \equiv (Z^\top Z)^{-1} X^\top Y$

then the CPU time is $O(np)$ and accuracy is as good as the usually estimator.
Fast Principal components regressions

Theorem (with Yichao Lu, Parmaveer Dhillion, Lyle Ungar)

If \( p > n \), then using a SRHT on the columns followed by regression will take \( O(np\log(p)) \) time and lose a constant factor on the statistical accuracy.
PCR is close to ridge regression

Theorem (with Sham Kakade, Parmaveer Dhillion, Lyle Ungar)

A ridge regression can be quickly approximated by regressing on the top principal components. In particular, for a ridge parameter $\lambda$ using components with singular values larger than $\lambda$ will be within a factor of 4 of the ridge estimator on statistical accuracy. (JMLR 2013)
Streaming feature selection was introduced in *JMLR* 2006 (with Zhou, Stine and Ungar).
Let $W(j)$ be the “alpha wealth” at time $j$. Then for a series of p-values $p_j$, we can define:

$$W(j) - W(j - 1) = \begin{cases} 
\omega & \text{if } p_j \leq \alpha_j, \\
-\alpha_j/(1 - \alpha_j) & \text{if } p_j > \alpha_j.
\end{cases} \quad (1)$$

**Theorem**

*(Foster and Stine, 2008, JRSS-B)* An alpha-investing rule governed by (1) with initial alpha-wealth $W(0) \leq \alpha \eta$ and pay-out $\omega \leq \alpha$ controls $mFDR_\eta$ at level $\alpha$. 

Theorem

(Foster, Dongyu Lin, 2011) VIF regression approximates a streaming feature selection method with speed $O(np)$. 
Eigenwords to estimate PERMA

See paper for the predictions of the other 4:

- **Positive emotion** (aglow, awesome, bliss, ...),
- **Engagement** (absorbed, attentive, busy, ...),
- **Relationships** (admiring, agreeable, ...),
- **Meaning** (aspire, belong, ...)
- **Achievement** (accomplish, achieve, attain, ...).

(P. Dhillon, J. Rodu, D. Foster and L. Ungar., ICML 2012)
(This is work in progress.) Yichao Lu has two current papers on this. The first shows how to use fast PCA and gradient decent to do a fast regression. The second shows how to use this successively to do a fast CCA. Kakade, Hsu and Zhang also have a fast CCA method, but it suffers from getting a less accurate answer than statistically optimal.
Theorem

(Foster, Johnson, Stine, 2013) If the R-squared in a regression is submodular (aka subadditive) then a streaming feature selection algorithm will find an estimator whose out risk is within a factor of $e/(e - 1)$ of the optimal risk.
This is the first theorem we did for HMMs. We now have many other versions for parsing and extensions to continuous data.

**Theorem (with Rodu, Ungar)**

Let \( X_t \) be generated by an \( m \geq 2 \) state HMM. Suppose we are given a \( U \) which has the property that \( \text{range}(O) \subset \text{range}(U) \) and \( |U_{ij}| \leq 1 \). Using \( N \) independent triples, we have

\[
N \geq \frac{128m^2(2t + 3)^2}{\epsilon^2 \Lambda^2 \sigma_m^4} \log \left( \frac{2m}{\delta} \right) \cdot \frac{1}{\epsilon^2/(2t + 3)^2} \left( \frac{2t+3}{\sqrt{1 + \epsilon - 1}} \right)^2
\]

implies that

\[
1 - \epsilon \leq \left| \frac{\hat{\Pr}(x_1, \ldots, x_t)}{\Pr(x_1, \ldots, x_t)} \right| \leq 1 + \epsilon
\]

holds with probability at least \( 1 - \delta \).
Theorem
Let $\hat{\beta}$ be the Ridge regression estimator with weights induced by the CCA. Then under the multi-view assumption

$$\text{Risk}(\hat{\beta}) \leq \left(5\alpha + \frac{\sum \lambda_i^2}{n}\right)\sigma^2$$
Theorem

Let $\hat{\beta}$ be the Ridge regression estimator with weights induced by the CCA. Then under the multi-view assumption

$$Risk(\hat{\beta}) \leq \left(5\alpha + \frac{\sum \lambda_i^2}{n}\right) \sigma^2$$

Estimator is least squares plus a penalty of:

$$\sum_i \frac{1 - \lambda_i}{\lambda_i} \beta_i^2$$

Where $\lambda_i$’s are the correlations
CCAs

**Theorem**

Let $\hat{\beta}$ be the Ridge regression estimator with weights induced by the CCA. Then under the multi-view assumption

$$\text{Risk}(\hat{\beta}) \leq \left(5\alpha + \frac{\sum \lambda_i^2}{n}\right) \sigma^2$$

Multiview property $\alpha$ is the multiview property:

$$\sigma_x^2 \leq \sigma_{x,z}^2 (1 + \alpha)$$
$$\sigma_z^2 \leq \sigma_{x,z}^2 (1 + \alpha)$$

- $5\alpha$ is the bias
- $\frac{\sum \lambda_i^2}{n}$ is variance
Results on 2 NLP sequence labeling problems: NER (CoNLL ’03 shared task) and Chunking (CoNLL ’00 shared task).

Trained on ∼ 65 million tokens of unlabeled text in a few hours!

Relative reduction in error over state-of-the-art:

<table>
<thead>
<tr>
<th>Embedding/Model</th>
<th>NER</th>
<th>Chunking</th>
</tr>
</thead>
<tbody>
<tr>
<td>C&amp;W</td>
<td>15.0%</td>
<td>18.8%</td>
</tr>
<tr>
<td>HLBL</td>
<td>19.5%</td>
<td>20.2%</td>
</tr>
<tr>
<td>Brown</td>
<td>12.1%</td>
<td>18.7%</td>
</tr>
<tr>
<td>Ando+Zhang</td>
<td>5.6%</td>
<td>14.6%</td>
</tr>
</tbody>
</table>

In EMNLP 2012 (Rodu, Ungar, Dhillon, Collins) we extended the HMM results to dependency parsing.
We have a review paper: “Spectral Learning of Latent-Variable PCFGs,” with Cohen, Stratos, Collins, and Ungar, submitting to *JMLR*.
Figure 1: Correlation of raw observations, binned at 10 second bins
Figure 2: Correlations among reduced dimensional observations $k=10$
Figure 3: Correlations among the states of the system as time progresses $k=10$