Talk 5: Linear methods for large data

Dean Foster

Amazon

- I meant to integrate Calibration in with the rest of this talk–but that didn't happen.
- So let me discuss one aspect of Calibration at Amazon.
- Switching to Chalk technology.

Preamble

- I meant to integrate Calibration in with the rest of this talk–but that didn't happen.
- So let me discuss one aspect of Calibration at Amazon.
- Switching to Chalk technology.
- Summary:
  - Main message: We need to do papers and theory to get good results!
  - Customers (in optimization) complained the forecasts were too variable
  - Bob and I looked at it and found that it was too variable (see black board) and wrote a paper about it (Arxiv 2021)
  - The NN team fixed it by adding an attention layer (Dhruv and gang, Arxiv 2021)
  - This avoided us needing to implement my LP to fix things (2019 conference paper).

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

- by Halko, Martinsson, and Tropp.
- It **USED TO BE** 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 $M \approx AA^\top M$

problem Find a low rank approximation to a $n \times m$ matrix M.
solution Find a $n \times k$ matrix A such that $M \approx AA^\top M$

Construction A is constructed by:

1. create a random $m \times k$ matrix Ω (iid normals)
2. compute $M\Omega$
3. Compute thin SVD of result: $UDV^\top = M\Omega$
4. $A = U$

FAST MATRIX REGRESSIONS

Toy problem: $p \ll n$:

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.

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" $\hat{X} \equiv A A^\top X$
- Estimate
$$\hat{\beta} = (\hat{X}^\top \hat{X})^{-1} X^\top Y$$
- (Mahoney also subsampled Y and hence lost accuracy.)

Fast Regressions

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

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.

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)

- If p is almost as large as n, then subsampling doesn't provide any speed up
    - To sub-sample enough to be accurate leaves the problem large
    - So best we could do would end up with a regularized answer

Outline:

1. Least squares regression (we just finished).
2. Sparse Linear Regression (up next).
3. Fast CCAs.
4. Fast HMMs.
5. Fast parsing.
6. Fast clustering.

   ⋮

   (Zeno will kick in somewhere in this series)

All are connected to the fast matrix decomposition.

(2) Sparse linear regression

- How do we do alpha investing fast?
- We need to estimate the t statistics of a new variable
  - The partial t-statistic is $t_{real} = t_{fast}/\sqrt{VIF}$.
  - We can compute the t-statistic for the simple regression $X_{new}^\top(Y - \hat{Y})$ quickly.
  - Compute the VIF by subsampling rows of data

Capacity
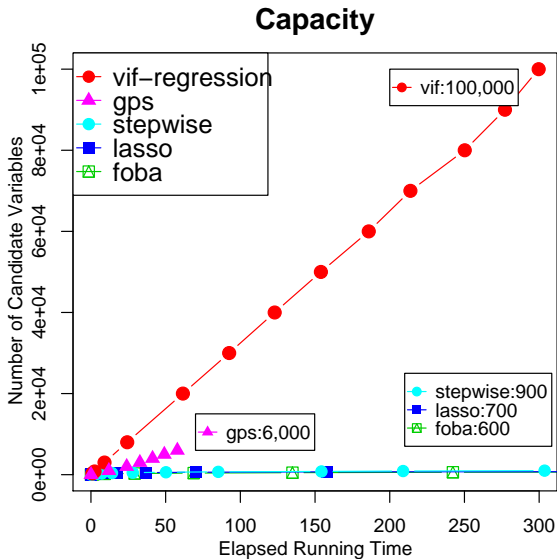
(3) CCA for Semi-supervised data

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

with $p \gg n$

m rows of unlabeled data:

$$
\begin{bmatrix} Y \\ n \times 1 \end{bmatrix}
\begin{bmatrix} X \\ (n+m) \times p \end{bmatrix}
$$

m rows of unlabeled data, and two sets of equally useful *X*'s:

$$
\begin{bmatrix} Y \\ n \times 1 \end{bmatrix}
\begin{bmatrix} X \\ (n+m) \times p \end{bmatrix}
\begin{bmatrix} Z \\ (n+m) \times p \end{bmatrix}
$$

With: $m \gg n$

Examples

- Named entity recognition
  - Y = person / place
  - X = spelling of the 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)

Theorem

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

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

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 λ_i's are the correlations

Theorem

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

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

Multivew property α is the multiview property:

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

- 5α is the bias
- $\frac{\sum \lambda_i^2}{n}$ is variance

Theorem

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

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

Results:

- Interpretation: Using the top few CCA directions is almost as good as the best linear model.
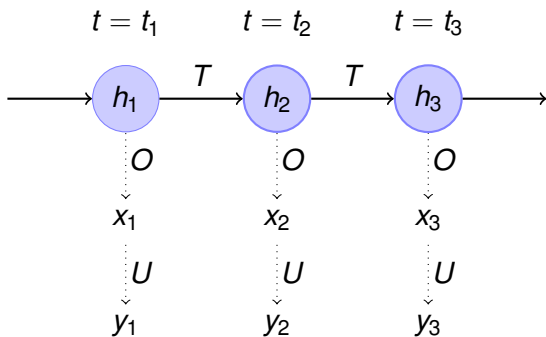- We can use this to generate Eigenwords (ICML 2012).
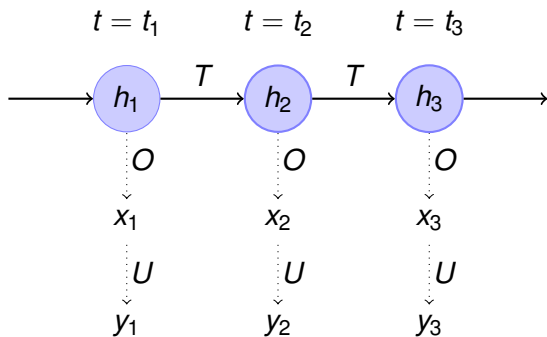
(4) HMMs

Hidden Markov Model
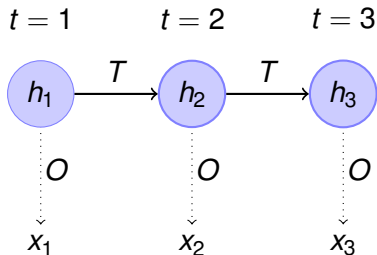
HMM with states h_1, h_2, and h_3 which generate observations x_1, x_2, and x_3.

Hidden Markov Model

The *Y*'s are our eigenwords.

Hidden Markov Model

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

The *Y*'s are our eigenwords.

- We called them "Eigenwords"

HMM 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
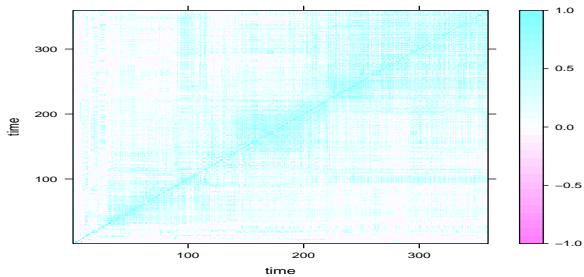 - (2011)
- Neural data (2013)

Figure 1: Correlation of raw observations, binned at 10 second bins
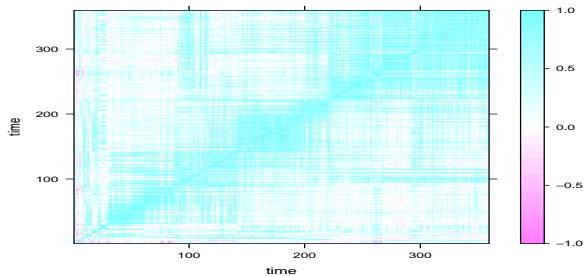
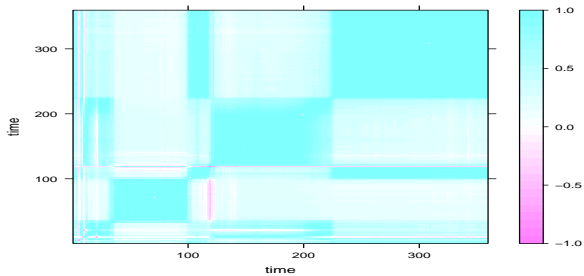Figure 2: Correlations among reduced dimensional observations k=10

Figure 3: Correlations among the states of the system as time progresses k=10

- (5) Extends to trees:

- (5) Extends to trees:
  - 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)

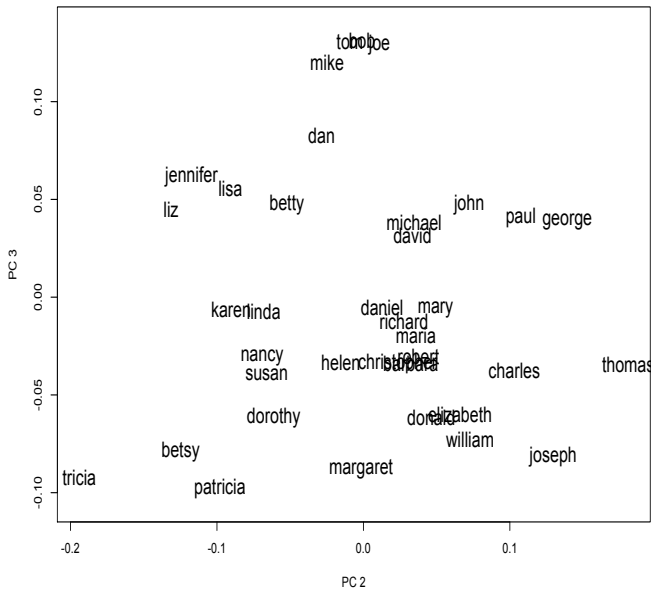**Theorem (with Hsu, Kakade, Liu, Anandkumar, NIPS 2012)**

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

Theorem (with Hsu, Kakade, Liu, Anandkumar, NIPS 2012)

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

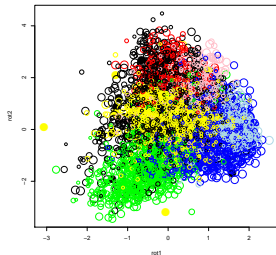- This allows clustering
- Don't get distracted by tensors! You can't really estimate them with real data.
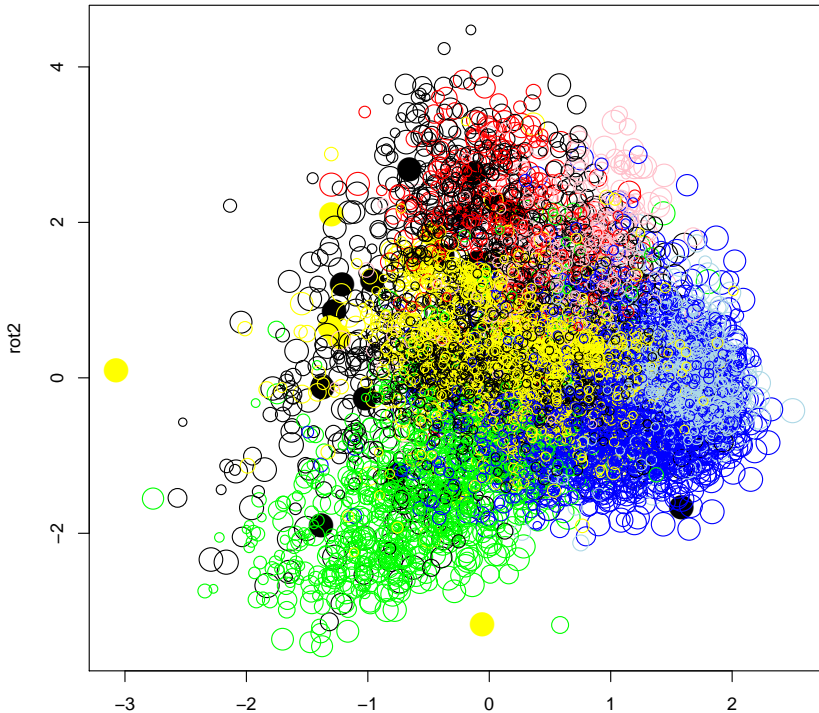
- These new fast matrix methods are easy to program.
- They generate statistically useful results.
- So, read Halko, Martinsson, and Tropp!
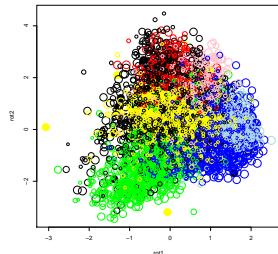
- These new fast matrix methods are easy to program.
- They generate statistically useful results.
- So, read Halko, Martinsson, and Tropp!

Thanks!

		U Penn	Elsewhere
Stat	faculty	Edward George Robert Stine	Tong Zhang (Rutgers) Sham Kakade (UW)
	students	Dongyu Lin (ATT) Jordan Rodu (CMU) Kory Johnson Yichao Lu (Amazon)	
CS	faculty	Lyle Ungar	Michael Collins (Columbia) Daniel Hsu (Columbia)
	students	Parmaveer Dhillon Jing Zhou (real world)	Shay Cohen (Columbia) Karl Stratos (Rutgers)

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.

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 λ using components with singular values larger than λ will be within a factor of 4 of the ridge estimator on statistical accuracy. (*JMLR *2013)*

Problem: Regress Y X, where X has $p \approx n$. Hence we can't simply subsample the X since it will still be too large.

1. Find a low rank (of dim k) approximation to X, call this matrix P.
2. Fit Y to this approximation and compute the residuals R
3. Run a gradient least squares regression of R on both P and X.

This method can be thought of as a preconditioned gradient method. Yichao Lu and I (2015) showed that this has good performance for a wide variety of singular values for X.

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:

$$W(j) - W(j-1) = \left\{ \begin{array}{ll} \omega & \text{if } p_j \leq \alpha_j, \\ -\alpha_j/(1-\alpha_j) & \text{if } p_j > \alpha_j. \end{array} \right. \tag{1}$$
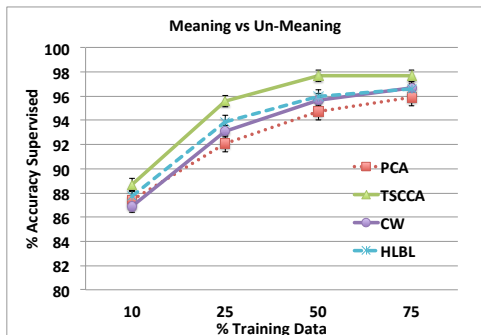
Theorem

(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 α.*

Theorem

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

Eigenwords to estimate PERMA

Meaning vs Un-Meaning

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.

HMM theorem

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 range(O) $\subset$ range(U) and $|U_{ij}| \leq 1$. Using N independent triples, we have*
>
> $$N \geq \frac{128m^2(2t+3)^2}{\epsilon^2\,\Lambda^2\sigma_m^4} \log\left(\frac{2m}{\delta}\right) \cdot \overbrace{\frac{\epsilon^2/(2t+3)^2}{(\sqrt[2t+3]{1+\epsilon}-1)^2}}^{\approx 1}$$
>
> *implies that*
>
> $$1-\epsilon \leq \left|\frac{\widehat{\Pr}(x_1,\ldots,x_t)}{\Pr(x_1,\ldots,x_t)}\right| \leq 1+\epsilon$$
>
> *holds with probability at least $1-\delta$.*

Results on ConLL task

- Results on 2 NLP sequence labeling problems: NER (CoNLL '03 shared task) and Chunking (CoNLL '00 shared task).
- Trained on $\sim$ 65 million tokens of unlabeled text in a few hours!

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

Embedding/Model	NER	Chunking
C&W	15.0%	18.8%
HLBL	19.5%	20.2%
Brown	12.1%	18.7%
Ando+Zhang	5.6%	14.6%

"Multi-View Learning of Word Embeddings via CCA," *NIPS* 2011.

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*.