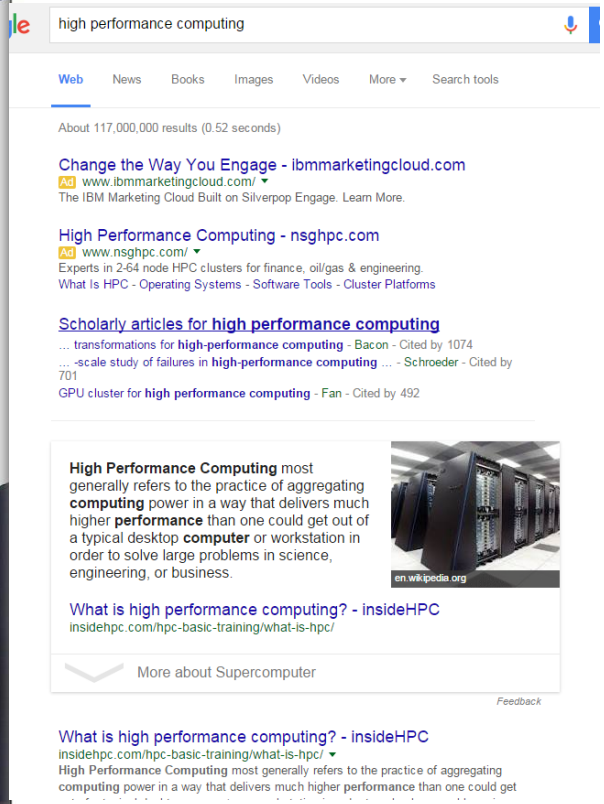
Networking & Information Technology Research and Development Program

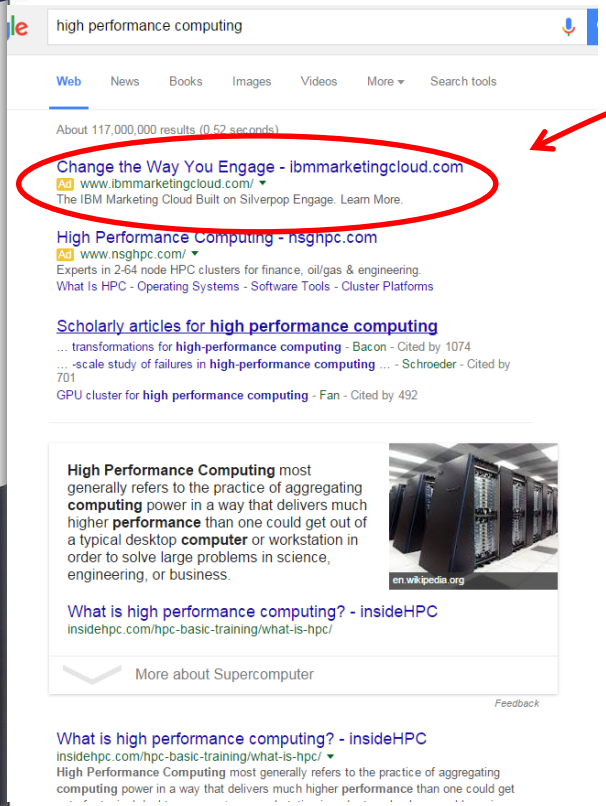
Andrew W. Moore (awm@cs.cmu.edu)
Carnegie Mellon, School of Computer Science
Formerly VP Engineering, Google

Good natured provocative remarks: It is, basically, a collision

A Big Data Machine Learning Problem



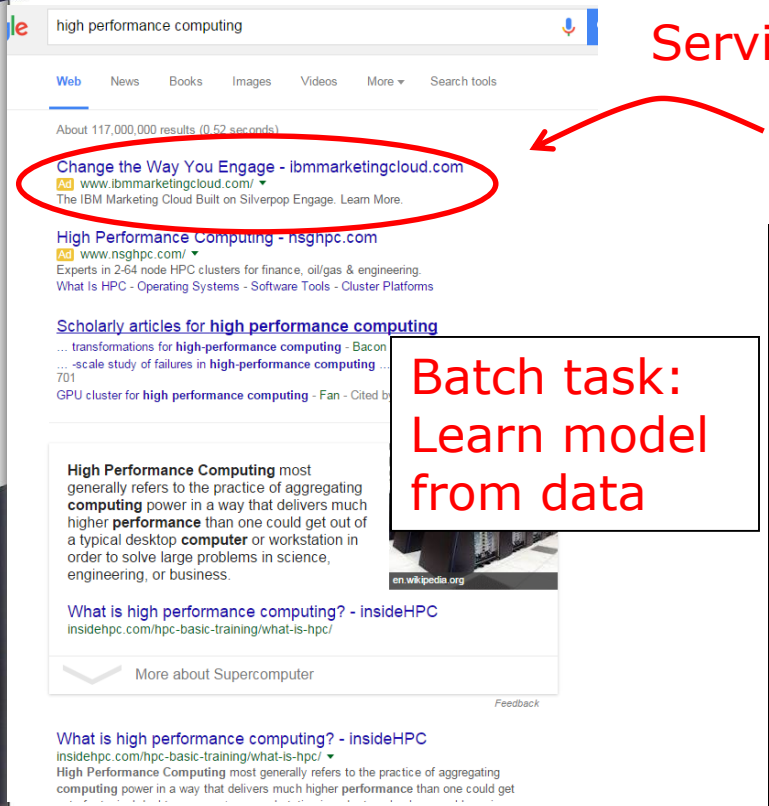
A Big Data Machine Learning Problem



Serving-time task:

$$P(\text{user clicks ad} \mid \text{query} = \text{"hpc"} \text{ and context})$$

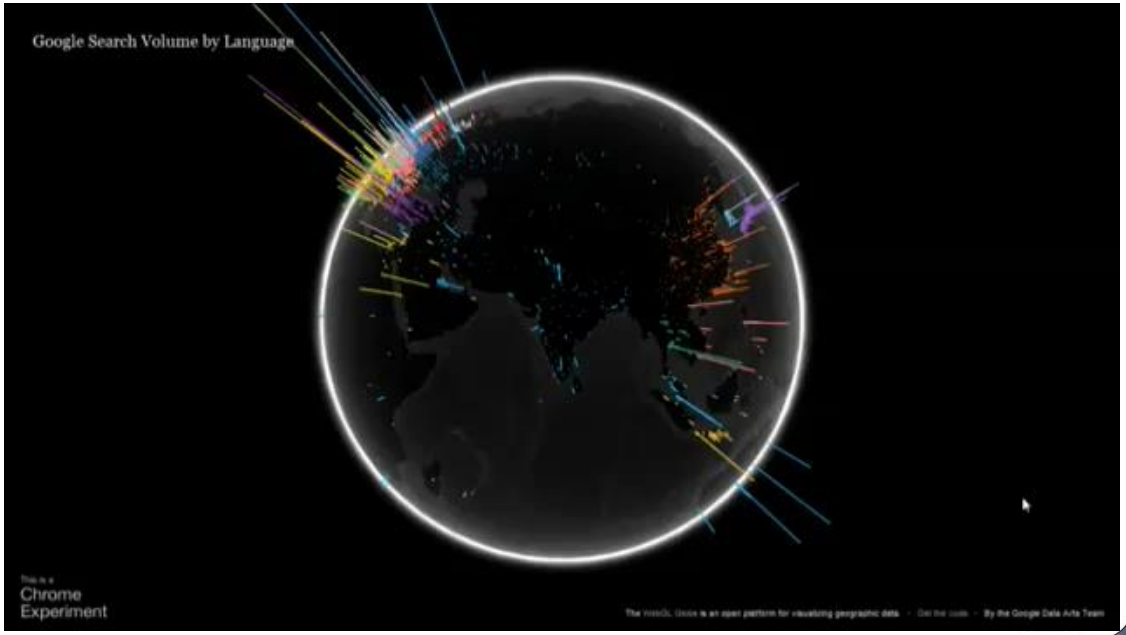
A Big Data Machine Learning Problem



Serving-time task:

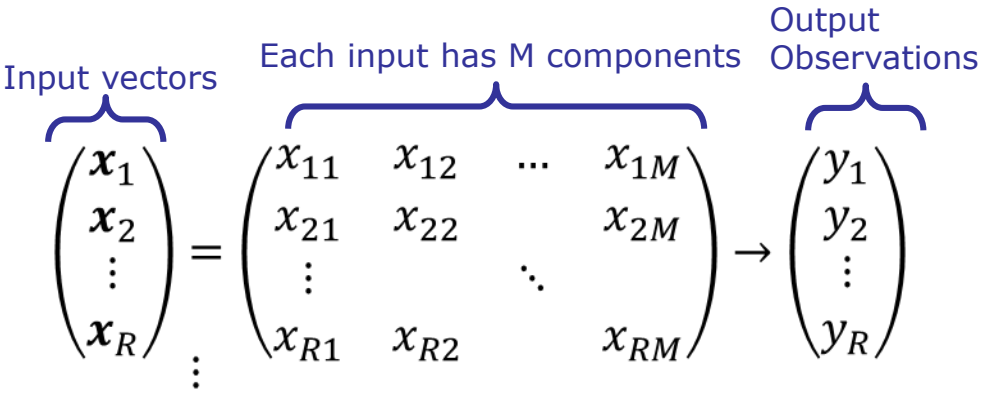
$$P(\text{user clicks ad} \mid \text{query} = \text{"hpc"} \text{ and context})$$

Batch task: Learn model from data



This is Machine Learning

Training Data consists of *R* records:



This is Machine Learning

Training Data consists of *R* records:

Input vectors

Each input has M components

Output Observations

$$\begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_R \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1M} \\ x_{21} & x_{22} & & x_{2M} \\ \vdots & & \ddots & \\ x_{R1} & x_{R2} & & x_{RM} \end{pmatrix} \rightarrow \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_R \end{pmatrix}$$

Learning Task:

Find a model represented by a set of *N* weights

$\boldsymbol{w} = w_1 , w_2 .. w_N$

which accurately predicts

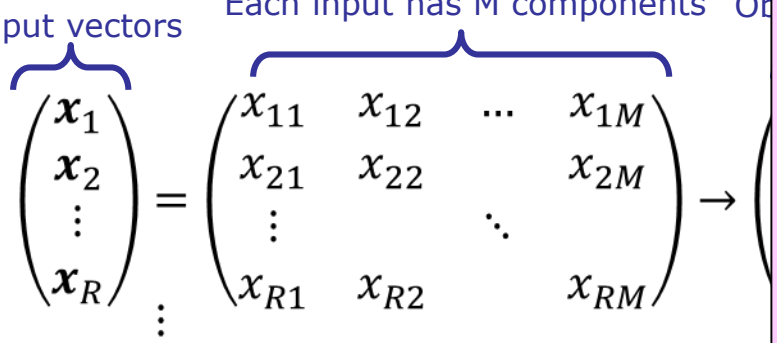
$P(y_{new} \mid \boldsymbol{x}_{new} , \boldsymbol{w})$

This is Machine Learning

Training Data consists of *R* records:

Input vectors

Each input has M components



$$\underset{w}{\operatorname{argmax}} P(\boldsymbol{w} | \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_R, y_1, y_2, \ldots y_R)$$

Learning Task:

Find a model represented by a set of *N* weights

$\boldsymbol{w} = w_1 , w_2 \,..\, w_N$

which accurately predicts

$P(y_{new} \mid \boldsymbol{x}_{new} , \boldsymbol{w})$

This is Machine Learning

Training Data consists of *R* records:

Input vectors

Each input has M components

$$\begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_R \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1M} \\ x_{21} & x_{22} & & x_{2M} \\ \vdots & & \ddots & \\ x_{R1} & x_{R2} & & x_{RM} \end{pmatrix} \rightarrow$$

$$\underset{w}{\operatorname{argmax}} P(\boldsymbol{w}|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_R, y_1, y_2, \ldots y_R)$$

$$=$$

$$\underset{w}{\operatorname{argmax}} \log P(\boldsymbol{w}) + \sum_{i=1}^{R} \log P(y_i|\boldsymbol{x}_i, \boldsymbol{w})$$

Learning Task:

Find a model represented by a set of *N* weights

$\boldsymbol{w} = w_1 , w_2 .. w_N$

which accurately predicts

$P(y_{new} \mid \boldsymbol{x}_{new} , \boldsymbol{w})$

This is Machine Learning

Training Data consists of *R* records:

Input vectors

Each input has M components

$$\begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_R \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1M} \\ x_{21} & x_{22} & & x_{2M} \\ \vdots & & \ddots & \\ x_{R1} & x_{R2} & & x_{RM} \end{pmatrix} \rightarrow$$

Learning Task:

Find a model represented by a set of *N* weights

$\boldsymbol{w} = w_1 , w_2 .. w_N$

which accurately predicts

$P(y_{new} \mid \boldsymbol{x}_{new} , \boldsymbol{w})$

$$\underset{w}{\operatorname{argmax}} P(\boldsymbol{w} | \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_R, y_1, y_2, \ldots y_R)$$

$$=$$

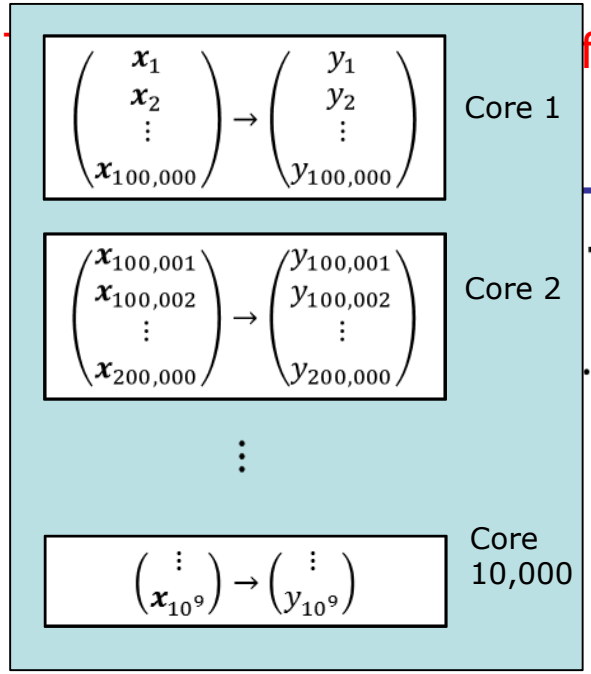
$$\underset{w}{\operatorname{argmax}} \log P(\boldsymbol{w}) + \sum_{i=1}^{R} \log P(y_i | \boldsymbol{x}_i, \boldsymbol{w})$$

If w_k is one of the current weights $\boldsymbol{w}$ then a better guess for w_k is

$$w_k \leftarrow w_k + h \frac{\frac{\partial}{\partial w_k} P(\boldsymbol{w})}{P(\boldsymbol{w})} + h \sum_{i=1}^{R} \frac{\frac{\partial}{\partial w_k} P(y_i | \boldsymbol{x}_i, \boldsymbol{w})}{P(y_i | \boldsymbol{x}_i, \boldsymbol{w})}$$

This is Machine Learning

f *R* records:



components

$$\begin{pmatrix} x_{1M} \\ x_{2M} \\ \\ x_{RM} \end{pmatrix} \rightarrow$$

$\boldsymbol{w} = w_1 , w_2 .. w_N$

which accurately predicts
$P(y_{new} \mid \boldsymbol{x}_{new} , \boldsymbol{w})$

$$\underset{w}{\operatorname{argmax}} P(\boldsymbol{w}|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_R, y_1, y_2, \ldots y_R)$$

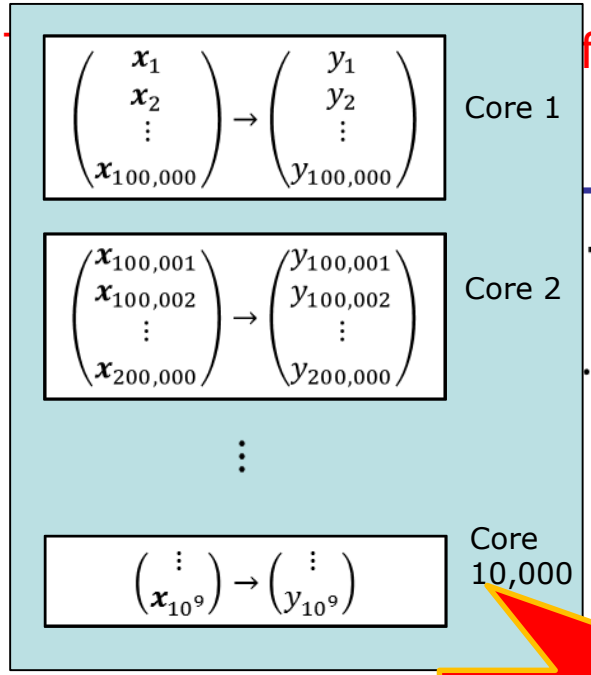
$$=$$

$$\underset{w}{\operatorname{argmax}} \log P(\boldsymbol{w}) + \sum_{i=1}^{R} \log P(y_i|\boldsymbol{x}_i, \boldsymbol{w})$$

If w_k is one of the current weights $\boldsymbol{w}$ then a better guess for w_k is

$$w_k \leftarrow w_k + h \frac{\frac{\partial}{\partial w_k} P(\boldsymbol{w})}{P(\boldsymbol{w})} + h \sum_{i=1}^{R} \frac{\frac{\partial}{\partial w_k} P(y_i|\boldsymbol{x}_i, \boldsymbol{w})}{P(y_i|\boldsymbol{x}_i, \boldsymbol{w})}$$

This is Machine Learning



f *R* records:

components

$$\begin{pmatrix} x_{1M} \\ x_{2M} \\ \\ x_{RM} \end{pmatrix} \rightarrow$$

$$\underset{w}{\operatorname{argmax}} P(\boldsymbol{w}|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_R, y_1, y_2, \ldots y_R)$$

$$=$$

$$\underset{w}{\operatorname{argmax}} \log P(\boldsymbol{w}) + \sum_{i=1}^{R} \log P(y_i|\boldsymbol{x}_i, \boldsymbol{w})$$

If w_k is one of the current weights $\boldsymbol{w}$ then a better guess for w_k is

$$\ldots \frac{\frac{\partial}{\partial w_k} P(\boldsymbol{w})}{P(\boldsymbol{w})} + h \sum_{i=1}^{R} \frac{\frac{\partial}{\partial w_k} P(y_i|\boldsymbol{x}_i, \boldsymbol{w})}{P(y_i|\boldsymbol{x}_i, \boldsymbol{w})}$$

$\boldsymbol{w} = w_1, w_2 \ldots$

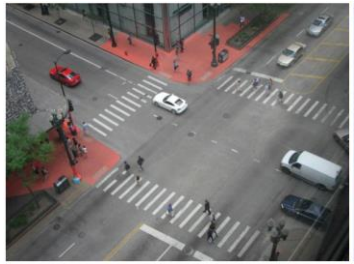
which accurately pred

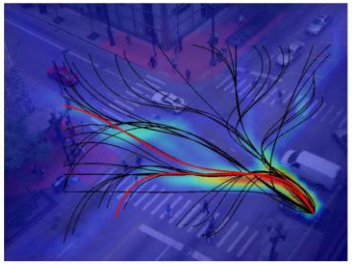
$P(y_{new} \mid \boldsymbol{x}_{new}, \ldots$

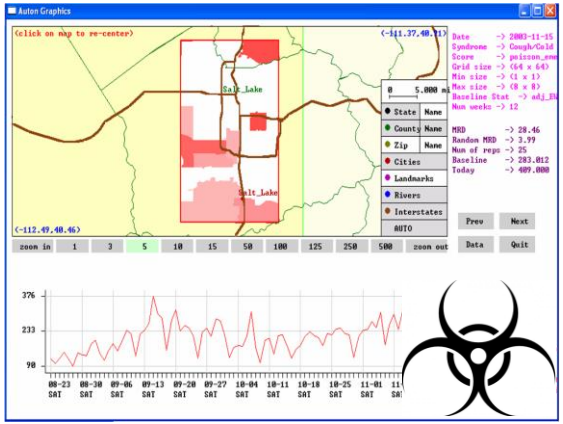
Very well suited to racks and racks of boring commodity servers with decent network, GPUs and flash

Use Cases for $R=10^{12}$ records, $M=10^{10}$ input dimensions

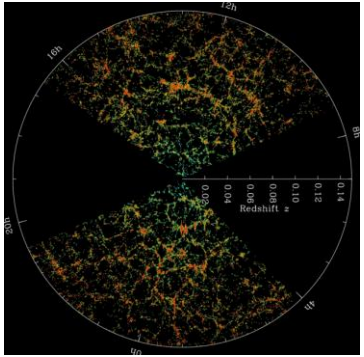
Google AdWords





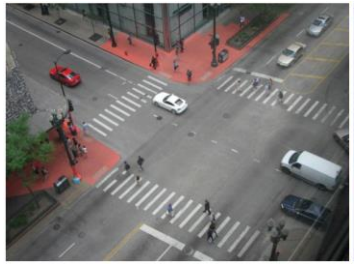


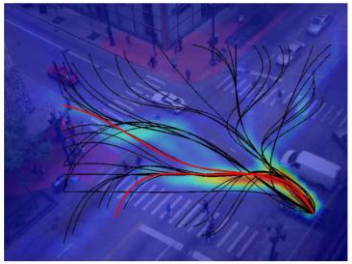
NETFLIX

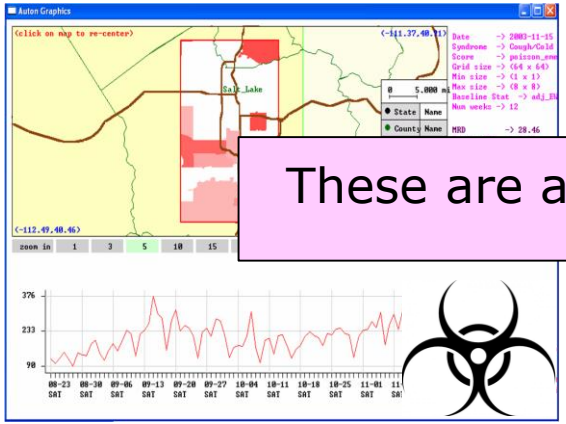


Use Cases for $R=10^{12}$ records, $M=10^{10}$ input dimensions

Google AdWords

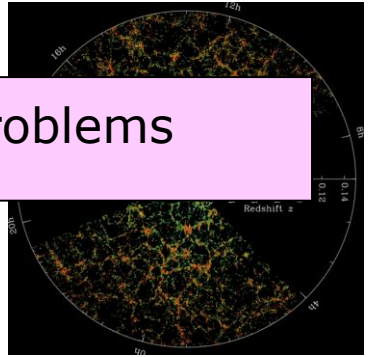




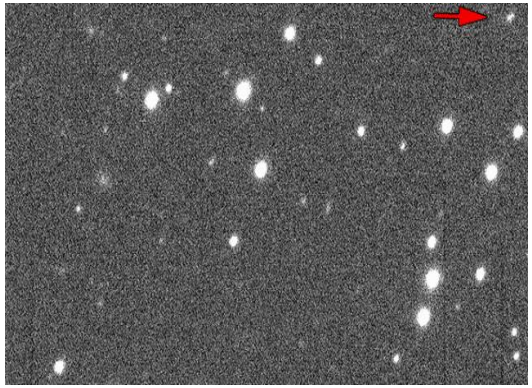


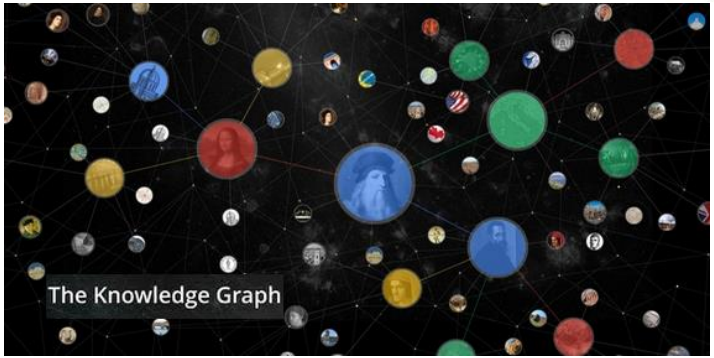
These are all $\underset{\boldsymbol{w}}{\operatorname{argmax}} P(\boldsymbol{w}|\text{data})$ problems

NETFLIX



But there are some with more of a multipole flavor





My Opinion

My gut feel on HPC and big data:

- Classic HPC is far removed from what is normally needed for big data
- But big data can use a lot of help with
  - Vector compute at nodes
  - Fast RAM cache over Flash
  - Does NOT need accurate RAM
  - Does NOT need reliable compute nodes
- Classic HPC will be much more important for the big-AI that will be built on the big-data

Questions, Comments, Ideas?
awm@cs.cmu.edu