Large-scale machine learning
Stochastic gradient descent

Mario Rodriguez

IRKM Lab

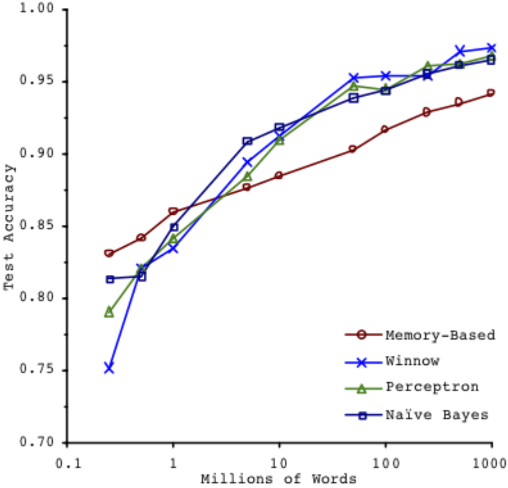
April 22, 2010

Introduction

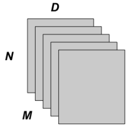
Very large amounts of data being generated quicker than we know what do with it ('08 stats):

- NYSE generates $\sim$ 1 terabyte/day of new trade data
- Facebook has about 1 billion photos $\sim$ 2.5 petabytes
- Large Hadron Collider $\sim$ 15 petabytes/year
- Internet Archive grows $\sim$ 20 terabytes/month

The dynamics of learning change with scale (Banko & Brill '01):



Dealing with large scale: challenges



Could be large:
N *(#data)*, ***D*** *(#features)*, ***M*** *(#models)*

- Data will most likely not fit in RAM
- Disk transfer speed slow compared to its size
  - $\sim$ 3 hrs to read 1 terabyte from disk @ 100MB/s
- 1 Gbit/s ethernet card (125 MB/s) reads about 450 GB/hour

Dealing with large scale: approaches

- Subsample from data available

Dealing with large scale: approaches

- Subsample from data available
- Reduce data dimensionality (compress, project to smaller subspace)

Dealing with large scale: approaches

- Subsample from data available
- Reduce data dimensionality (compress, project to smaller subspace)
- Implicit kernel representation (kernel trick, up to infinite dimensional spaces)

Dealing with large scale: approaches

- Subsample from data available
- Reduce data dimensionality (compress, project to smaller subspace)
- Implicit kernel representation (kernel trick, up to infinite dimensional spaces)
- Use smarter (quicker) algorithm (batch vs conjugate gradient descent)

Dealing with large scale: approaches

- Subsample from data available
- Reduce data dimensionality (compress, project to smaller subspace)
- Implicit kernel representation (kernel trick, up to infinite dimensional spaces)
- Use smarter (quicker) algorithm (batch vs conjugate gradient descent)
- Parallelize the algorithm (map-reduce, GPU, MPI, SAN-based HPC)

Dealing with large scale: approaches

- Subsample from data available
- Reduce data dimensionality (compress, project to smaller subspace)
- Implicit kernel representation (kernel trick, up to infinite dimensional spaces)
- Use smarter (quicker) algorithm (batch vs conjugate gradient descent)
- Parallelize the algorithm (map-reduce, GPU, MPI, SAN-based HPC)
- Use an incremental/stream-based algorithm (online learning, stochastic gradient descent)

Dealing with large scale: approaches

- Subsample from data available
- Reduce data dimensionality (compress, project to smaller subspace)
- Implicit kernel representation (kernel trick, up to infinite dimensional spaces)
- Use smarter (quicker) algorithm (batch vs conjugate gradient descent)
- Parallelize the algorithm (map-reduce, GPU, MPI, SAN-based HPC)
- Use an incremental/stream-based algorithm (online learning, stochastic gradient descent)
- Mix and match!

Review: batch gradient descent

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2.$$

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta).$$

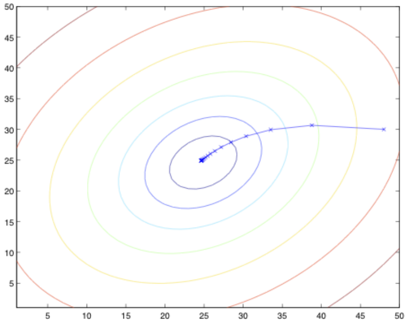
$$\begin{aligned}
\frac{\partial}{\partial\theta_j}J(\theta) &= \frac{\partial}{\partial\theta_j}\frac{1}{2}\left(h_\theta(x) - y\right)^2 \\
&= 2\cdot\frac{1}{2}\left(h_\theta(x) - y\right)\cdot\frac{\partial}{\partial\theta_j}\left(h_\theta(x) - y\right) \\
&= \left(h_\theta(x) - y\right)\cdot\frac{\partial}{\partial\theta_j}\left(\sum_{i=0}^{n}\theta_i x_i - y\right) \\
&= \left(h_\theta(x) - y\right)x_j
\end{aligned}$$

Review: batch gradient descent

Repeat until convergence {

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)} \qquad \text{(for every } j\text{)}.$$

}



Gradient descent

Loop {

for i=1 to m, {

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)} \qquad \text{(for every } j\text{)}.$$

}

}

Algorithm: Stochastic gradient descent

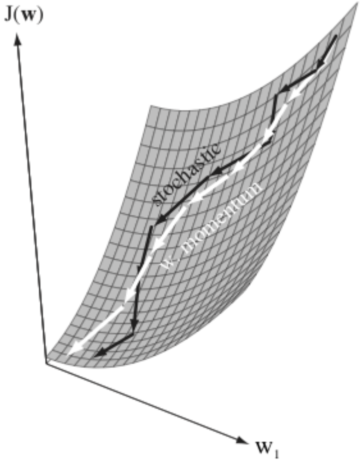
Repeat until convergence {

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)} \qquad \text{(for every } j\text{)}.$$

}

Algorithm: Batch gradient descent

Stochastic gradient descent: what does it look like?



Stochastic gradient descent: why does it work?

$$
\begin{aligned}
\boldsymbol{\mu}_{\mathrm{ML}}^{(N)} &= \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n \\
&= \frac{1}{N}\mathbf{x}_N + \frac{1}{N}\sum_{n=1}^{N-1}\mathbf{x}_n \\
&= \frac{1}{N}\mathbf{x}_N + \frac{N-1}{N}\boldsymbol{\mu}_{\mathrm{ML}}^{(N-1)} \\
&= \boldsymbol{\mu}_{\mathrm{ML}}^{(N-1)} + \frac{1}{N}(\mathbf{x}_N - \boldsymbol{\mu}_{\mathrm{ML}}^{(N-1)}).
\end{aligned}
$$

Stochastic gradient descent: why does it work?

- Foundational work in stochastic approximation methods by Robins and Monro, in the 1950's
- These algorithms have proven convergence in the limit, as the number of data points goes to infinity, provided a few things hold:

$$\theta^{(N)} = \theta^{(N-1)} + a_{N-1} z(\theta^{(N-1)})$$

$$\begin{aligned} \lim_{N\to\infty} a_N &= 0 \\ \sum_{N=1}^{\infty} a_N &= \infty \\ \sum_{N=1}^{\infty} a_N^2 &< \infty. \end{aligned}$$

Stochastic gradient descent: why should we care?

- In practice, stochastic gradient descent will often get close to the minimum much faster than batch gradient descent

Stochastic gradient descent: why should we care?

- In practice, stochastic gradient descent will often get close to the minimum much faster than batch gradient descent
- In addition, with an unlimited supply of data, stochastic gradient descent is the obvious candidate

Stochastic gradient descent: why should we care?

- In practice, stochastic gradient descent will often get close to the minimum much faster than batch gradient descent
- In addition, with an unlimited supply of data, stochastic gradient descent is the obvious candidate
- Bottou and Le Cun (2003) show that the best generalization error is asymptotically achieved by the learning algorithm that uses the **most examples within the allowed time**

Small-scale vs large-scale (Bottou & Bousquet '07)

$$
\begin{aligned}
E(\tilde{f}_n) - E(f^*) &= E(f^*_{\mathcal{F}}) - E(f^*) && \text{Approximation error} \\
&+ E(f_n) - E(f^*_{\mathcal{F}}) && \text{Estimation error} \\
&+ E(\tilde{f}_n) - E(f_n) && \text{Optimization error}
\end{aligned}
$$

Problem: Choose $\mathcal{F}$, n, ρ to make error as small as possible, subject to constraints:

- Large scale: constraint is computation time
- Small scale: constraint is number of examples

Small-scale vs large-scale (Bottou & Bousquet '07)

$$\begin{aligned} E(\tilde{f}_n) - E(f^*) &= E(f^*_{\mathcal{F}}) - E(f^*) && \text{Approximation error} \\ &+ E(f_n) - E(f^*_{\mathcal{F}}) && \text{Estimation error} \\ &+ E(\tilde{f}_n) - E(f_n) && \text{Optimization error} \end{aligned}$$

Approximation error bound:
– decreases when $\mathcal{F}$ gets larger.

Estimation error bound:
– decreases when n gets larger.
– increases when $\mathcal{F}$ gets larger.

Optimization error bound:
– increases with ρ.

Computing time T:
– decreases with ρ
– increases with n
– increases with $\mathcal{F}$

Small-scale vs large-scale (Bottou & Bousquet ’07)

- Small scale
  - To reduce estimation error use as many examples as possible
  - To reduce optimization error take $\rho = 0$
  - Adjust richness of $\mathcal{F}$

Small-scale vs large-scale (Bottou & Bousquet '07)

- Small scale
  - To reduce estimation error use as many examples as possible
  - To reduce optimization error take $\rho = 0$
  - Adjust richness of $\mathcal{F}$
- Large scale
  - More complicated: computing time depends on all 3 parameters ($\mathcal{F}$, n, ρ)
  - Example: If we choose ρ small, we decrease the optimization error, but we may then also have to decrease $\mathcal{F}$ and/or n, with adverse effects on the estimation and approximation errors

Algorithmic complexity (Bottou '08)

	Cost per iteration	Iterations to reach ρ	Time to reach accuracy ρ
GD	$\mathcal{O}(nd)$	$\mathcal{O}\left(\kappa \log \frac{1}{\rho}\right)$	$\mathcal{O}\left(nd\kappa \log \frac{1}{\rho}\right)$

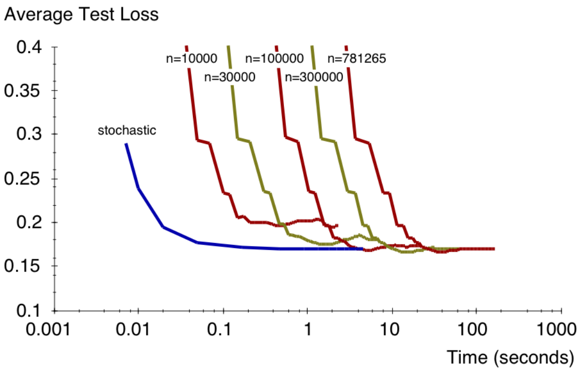
	Cost per iteration	Iterations to reach ρ	Time to reach accuracy ρ
SGD	$\mathcal{O}(d)$	$\frac{\nu k}{\rho} + o\left(\frac{1}{\rho}\right)$	$\mathcal{O}\left(\frac{d \nu k}{\rho}\right)$

Representative results (Bottou '08)

- **Dataset**
  - Reuters RCV1 document corpus.
  - 781,265 training examples, 23,149 testing examples.
  - 47,152 TF-IDF features.

	Training Time	Primal cost	Test Error
SVMLight	23,642 secs	*0.2275*	*6.02%*
SVMPerf	66 secs	0.2278	6.03%
SGD	1.4 secs	*0.2275*	*6.02%*

Representative results (Bottou '08)



Log-loss problem

Batch Conjugate Gradient on various training set sizes

Stochastic Gradient on the full set

Stochastic gradient descent: usage

- Least squares regression

- Least squares regression
- Generalized linear models

Stochastic gradient descent: usage

- Least squares regression
- Generalized linear models
- Multilayer neural networks

Stochastic gradient descent: usage

- Least squares regression
- Generalized linear models
- Multilayer neural networks
- Singular value decomposition

Stochastic gradient descent: usage

- Least squares regression
- Generalized linear models
- Multilayer neural networks
- Singular value decomposition
- Principal component analysis

Stochastic gradient descent: usage

- Least squares regression
- Generalized linear models
- Multilayer neural networks
- Singular value decomposition
- Principal component analysis
- Support vector machines
  - Pegasos: Primal Estimated sub-GrAdient SOlver for SVM (Shalev-Shwartz et al. '07)

Stochastic gradient descent: usage

- Least squares regression
- Generalized linear models
- Multilayer neural networks
- Singular value decomposition
- Principal component analysis
- Support vector machines
  - Pegasos: Primal Estimated sub-GrAdient SOlver for SVM (Shalev-Shwartz et al. '07)
- Conditional random fields
  - Accelerated Training of Conditional Random Fields with Stochastic Gradient Methods (Vishwanathan et al. '06)

Stochastic gradient descent: usage

- Least squares regression
- Generalized linear models
- Multilayer neural networks
- Singular value decomposition
- Principal component analysis
- Support vector machines
  - Pegasos: Primal Estimated sub-GrAdient SOlver for SVM (Shalev-Shwartz et al. '07)
- Conditional random fields
  - Accelerated Training of Conditional Random Fields with Stochastic Gradient Methods (Vishwanathan et al. '06)
- Code and results for the last 2 @ http://leon.bottou.org/projects/sgd

Stochastic gradient descent: how to speed/scale it up?

- Second-order stochastic gradient

Stochastic gradient descent: how to speed/scale it up?

- Second-order stochastic gradient
- Mini-batches

Stochastic gradient descent: how to speed/scale it up?

- Second-order stochastic gradient
- Mini-batches
- Parallelize over examples

Stochastic gradient descent: how to speed/scale it up?

- Second-order stochastic gradient
- Mini-batches
- Parallelize over examples
- Parallelize over features

Stochastic gradient descent: how to speed/scale it up?

- Second-order stochastic gradient
- Mini-batches
- Parallelize over examples
- Parallelize over features
- Vowpal Wabbit @ http://hunch.net/∼vw/

Conclusion

- Stochastic gradient descent:
  - Simple
  - Fast
  - Scalable

Conclusion

- Stochastic gradient descent:
  - Simple
  - Fast
  - Scalable
- Don't underestimate it!