Fast Clustering leads to Fast SVM Training and More

Daniel Boley

University of Minnesota

Supported in part by NSF
Goals and Outline

• Existence of Fast Clustering methods makes possible several applications.
  ◦ Compare deterministic and non-deterministic clusterers.

• Fast training of Support Vector Machines.

• Low Memory Factored Representation, for data too big to fit in memory.
  ◦ Fast clustering of datasets too big to fit in memory.
  ◦ Fast generalization of LSI for document retrieval.
  ◦ Representation of Streaming Data.
Hierarchical Clustering

- Clustering at all levels of resolution.
- Bottom-up clustering is $O(n^2)$.
- Top-down clustering can be made $O(n)$.
- Leads to PDDP. [basis of this talk].
Hierarchical Clustering: Get a Tree

Diagram showing a hierarchical clustering tree with nodes labeled "affirm", "patent", "technologi", "action", "employe", "employe...", "technologi", "employe...", "system", "develop", "manufactur...", "affirm", "action", "employe", "employe...", "technologi", "employe...", "system", "manufactur", "engin", "process...".
K-means: Popular Fast Clustering

- Quality of final result depends on initialization
- Random initialization \( \Rightarrow \) results hard to repeat.
- Deterministic initialization - no universal strategy
- Cost: \( O(\text{#iters} \cdot m \cdot n) \Rightarrow \text{linear in } n. \)

where \( n = \text{number of data samples} \)

\( m = \text{number of attributes per sample}. \)
Simple Model

- Reduce to 1 parameter: angle $\alpha$.
- Major axis = 1, Minor axis = $a < 1$.
- Non-linear dynamic system: $\alpha_{t+1} = \arctan[a^2 \tan \alpha_t]$.
- # iterations to converge: $\approx -1/\log a^2$. 
Infinitely Many Points

K-means modelled as a fixed point iteration
Finite Number of Points

Number of data points = 15; $a=0.6$

Equilibrium points

Number of data points = 100; $a=0.6$
Finite Number of Points

- Many equilibrium points $\implies$ many local minima.
- As number of points grows, local minima tend to vanish.
- As minor axis $\to 1$, more local minima tend to appear.
PDDP vs K-means on Model Problem

- In the limit, PDDP & K-means yield same split here. [Savaresi]
Starting K-means

- Empirically, PDDP is a good seed for K-means.
Cost of K-means vs PDDP

- Both are linear in the number of samples.
- K-means often cheapest, but cost can vary a lot.
SVM via Clustering

- Motivation: Reduce training cost by clustering and use one representative per cluster instead of all the original data.

- Empirically provides good SVMs with comparable error rates on test sets.

- Theoretically generalization error satisfies “same” bound as the SVM obtained using all the data.

- Can be made adaptable by quickly running a sequence of SVMs, each with new data points added, to adjust and improve SVM adaptively.
SVM via Clustering

- Cluster Training Set into partitions
- Train SVM using 1 representative per partition.

\[ d(x) = \begin{cases} 1 & \text{if } x \in D_{pos}, 1 \\ > 1 & \text{if } x \in D_{pos}, 2 \\ < -1 & \text{if } x \in D_{neg}, 1 \\ > -1 & \text{if } x \in D_{neg}, 2 \\ < 1 & \text{if } x \in D_{neg}, 3 \\ \end{cases} \]

Decision boundary: \[ d(x) = \begin{cases} 1 & \text{if } x \in D_{pos}, 1 \\ < 1 & \text{if } x \in D_{pos}, 2 \\ > -1 & \text{if } x \in D_{neg}, 1 \\ < -1 & \text{if } x \in D_{neg}, 2 \\ > 1 & \text{if } x \in D_{neg}, 3 \\ \end{cases} \]
Support Vector Machine

- Minimize $R(d; \mathcal{D}, \lambda) = R_{\text{emp}}(d; \mathcal{D}) + \lambda \cdot \Omega(d)$

  - $R_{\text{emp}}(d; \mathcal{D})$: empirical error
  - $\Omega(d)$: regularization/complexity term

- $\mathcal{D} = \{x_i, y_i\}_{i=1}^{n}$: training set.
- $x_i$: datum w/ label $y_i = \pm 1$.
- $\phi(x)$: non-linear lifting.
- $d(x) = \langle w, \phi(x) \rangle$: discriminant fcn.
- $\lambda$: regularization coefficient
- $\Omega(d) = \|w\|^2$

- $R_{\text{emp}}(d; \mathcal{D}) = \frac{1}{n} \sum_{(x, y) \in \mathcal{D}} \ell_{\text{hinge}}(d, (x, y)) = \max\{0, 1 - y \cdot d(x)\}$
Questions to be Resolved

• How to select representatives?

• If selection cost is $O(n^2)$
  then one gains little by using representatives.

• How to adjust representatives to improve classifier quality?
Approximate SVM Methods

Choices of Clustering Method

- Use fast clustering method.
- Intuition: want to minimize distance sample point ⇔ representative in lifted space.
- $\rightarrow$ kernel K-means.
- But expensive, so approximate it with
  - data K-means (natural choice)
  - data PDDP (to make deterministic or to init K-means)
- Option: add potential support vectors, and repeat.
Quality of SVM – Theory

• Could apply VC dimension bounds, but we want something tighter.

• Extend Algorithmic-Stability bounds to this case. These apply specifically to learning algorithms minimizing some convex functional, whose change is bounded when a datum is substituted.

• Assume only that representatives are centers of partitions.

• Partitions are arbitrary, so result applies even when using data K-means, data space PDDP, random partitioning, or even a sub-optimal soln from kernel K-means.
Stability Bound Theorem

Get theorem much like one for Exact SVM.

- For any $n \geq 1$ and $\delta \in (0, 1)$, with confidence at least $1 - \delta$ over the random draw of a training data set $\mathcal{D}$ of size $n$:

$$
\mathbb{E}(\mathbb{I}_{\tilde{h}(x) \neq y}) \leq \frac{1}{n} \sum_{(x,y) \in \mathcal{D}} \ell_{\text{hinge}}(\tilde{h}, x, y) + \frac{\chi^2}{\lambda n} + \left( \frac{2\chi^2}{\lambda} + 1 \right) \sqrt{\frac{\ln 1/\delta}{2n}}.
$$

where

- $\tilde{h}(x) \overset{\text{def}}{=} \text{sign} \{ \tilde{d}(x) \}$ is the approximate SVM.
- $\chi^2 = \max_i K(x_i, x_i) = \max \langle \phi(x_i), \phi(x_i) \rangle$ (1 for RBF kernel).
- $\lambda$ corresponds to soft-margin weighting. trade-off of training error $\leftrightarrow$ sensitivity.
Experimental Setup

• Illustrate performance of SVM with clustering on some examples.

• We cluster in data space with PDDP;

• We compare the proposed algorithm against the standard training algorithm SMO [Platt, 1999], implemented in LibSVM [Chang+Lin 2001] [Fan 2005];
## Experimental Performance

<table>
<thead>
<tr>
<th>Data set (Size)</th>
<th>Exact SVM</th>
<th>Approximate SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$T_{\text{train}}$ (sec.)</td>
<td>Accuracy</td>
</tr>
<tr>
<td>UCI-Adult (32,561)</td>
<td>1,877</td>
<td>95.7%</td>
</tr>
<tr>
<td>UCI-Web (49,749)</td>
<td>2,908</td>
<td>99.8%</td>
</tr>
<tr>
<td>MNIST (60,000)</td>
<td>6,718</td>
<td>98.8%</td>
</tr>
<tr>
<td>Yahoo (100,000)</td>
<td>18,437</td>
<td>83.8%</td>
</tr>
</tbody>
</table>
Low Memory Factored Representation

- Use clustering to construct a representation of a full massively large data sets in much less space.
- Representation is not exact, but every individual sample has its own unique representative in the approximate representation.
- In principle, would still allow detection and analysis of outliers and other unusual individual samples.
- Next slide has basic idea.
Low Memory Factored Representation

\[ \mathbf{M} \]

\[ n \quad 1 \quad 2 \quad \ldots \quad k_d \quad \ldots \quad k_s \]

Clustering

Least Squares

\[ \mathbf{C} \]

\[ k_c \]

section representatives

\[ \mathbf{Z} \]

very sparse

\[ k_c \quad k_z \text{ nonzeros per column} \]

data loadings

\[ m \quad \cdot \quad m \]

\[ \mathbf{Z} \]
Fast factored representation: LMFR

[Littau]

\[ M = CZ \text{ by fast clustering of each section} \]

\[ C = \text{matrix of representatives} \]

Still have \( Z \) to individualize representation of each sample

\[ \text{Make } Z \text{ sparse to save space}. \]

linear clustering cost \( \rightarrow \) linear cost to construct LMFR

In principle, could use any fast clusterer.

We use PDDP to make it more deterministic.
LMFR ⇒ Clustering ⇒ PMPDDP

Using PDDP on an LMFR yields Piece-Meal PDDP.

• Factored Representation ⇒ to reconstruct data

• Expensive to compute similarities between individual data.

• Want to avoid accessing individual data.

• Ideal for clusterer that depends on $M \times v$’s

• A spectral clustering method like PDDP is a good fit.

• Experimentally, cluster quality $\approx$ plain PDDP.
⇒ **PMPDDP - Piece-Meal PDDP**

- Divide original data $\mathbf{M}$ up into sections
  Extract representatives for each section, fast.
  [can be imperfect]

- Matrix of representatives $\Rightarrow \mathbf{C}$

- Approximate each original sample as a linear combination of $k$ representatives [selected via least squares].

- Matrix of coefficients $\Rightarrow \mathbf{Z}$

- $k$ is a small number like 3 or 5.

- Apply PDDP to the product $\mathbf{CZ}$ instead of original $\mathbf{M}$. [never multiply out $\mathbf{CZ}$ explicitly]
PMPDDP – on KDD dataset

- Still Linear in size of data set.
PMPDDP – on KDD dataset

- First 5 samples: PMPDDP cost \( \approx 4 \times \text{PDDP} \).
PMPDDP – on KDD dataset

- Memory usage small.

![Graph showing memory consumption vs number of samples. The graph plots memory usage in MB against the number of samples, illustrating a linear relationship.](image-url)
LMFR for Document Retrieval

- Mimic LSI, except we use factored representation $CZ$.
- Different from finding nearest concepts (ignoring $Z$)
- Can handle much larger datasets than Concept Decomposition [full $Z$]
- Less time needed to achieve similar retrieval accuracy.
Doc Retrieval Experiments

- Compare methods achieving similar retrieval accuracy.

<table>
<thead>
<tr>
<th>method</th>
<th>$k_c$</th>
<th>$k_z$</th>
<th>MB</th>
<th>sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>N.A.</td>
<td>N.A.</td>
<td>18.34</td>
<td>N.A.</td>
</tr>
<tr>
<td>rank 100 SVD</td>
<td>N.A.</td>
<td>N.A.</td>
<td>40.12</td>
<td>438</td>
</tr>
<tr>
<td>rank 200 concept decomposition</td>
<td>200</td>
<td>200</td>
<td>25.88</td>
<td>10294</td>
</tr>
<tr>
<td>LMFR</td>
<td>200</td>
<td>5</td>
<td>8.10</td>
<td>185</td>
</tr>
<tr>
<td>LMFR</td>
<td>300</td>
<td>5</td>
<td>9.17</td>
<td>188</td>
</tr>
<tr>
<td>LMFR</td>
<td>400</td>
<td>5</td>
<td>10.02</td>
<td>187</td>
</tr>
<tr>
<td>LMFR</td>
<td>500</td>
<td>5</td>
<td>10.68</td>
<td>189</td>
</tr>
<tr>
<td>LMFR</td>
<td>600</td>
<td>5</td>
<td>11.32</td>
<td>187</td>
</tr>
</tbody>
</table>
Doc Retrieval Experiments

Recall vs precision for the original representation $M$

Recall vs precision for the rank 100 SVD

Recall vs precision for the rank 200 concept decomposition

Recall vs precision for the LMFR, $k_c=600$, $k_z=5$
LMFR for Streaming Data

- Simple idea: collect data into sections as they arrive
- Form CZ section by section as they fill.
- Get LMFR for data, useful for any application (clustering, IR, aggregate statistics,...]
- No need to decide application in advance
LMFR for Streaming Data

- Memory for $Z$ grows very slowly.
- Memory for $C$ grows more.
- Recursively factor $C$ into its own $\hat{C}\hat{Z}$ ⇒ less space.
- Hybrid Approach: once in a while do a completely new LMFR.
Streaming Data Results

Memory used for 3 Update Methods for the KDD data

- rebuild CZ
- factor C
- hybrid

number of data items vs. memory occupied by $C_G Z_G$, in MB.
Streaming Data Results

Time Taken per data item for 3 Update Methods for the KDD data

- rebuild CZ
- factor C
- hybrid

number of data items vs. time per data item to compute $C_{Z,G}$ in seconds
Related Work

- **SVM via Clustering**
  - Chunking (Boser+92, Osuna+97, Kaufman+99, Joachims99)
  - Low Rank Approx (Fine 01, Jordan)
  - Sampling (Williams+Seeger01, Achlioptas+McSherry+Schölkopf 02)
  - Squashing (Pavlov+Chudova+Smith 00)
  - Clustering (Cao+04, Yu+Yang+Han 03)

- **Agglomeration on large datasets**
  - gather/scatter (Cutting+ 92)
  - CURE(Guha+98)
  - gaussian model (Fraley 99)
  - Heap (Kurita 91)
  - refinement (Karypis 99)
Related Work

- **K-means on large datasets**
  - Initialization (Bradley-Fayyad 1998)
  - kd-tree (Pelleg-Moore 1999)
  - Sampling (Domingos+01)
  - CLARANS k-medoid, spatial data (Ng+Han 94)
  - Birch (more sampling than k=means) (Ramakrishnan+96)

- **Matrix Factorization**
  - LSI Berry 95 Deerwester 90
  - Sparse LowRankApprox Zhang+Zha+Simon 2002
  - SDD (Kolda+98) – good for outlier detection (Skillikorn+01)
  - Monte-Carlo sampling (Vempala+98)
  - Concept Decomp (Dhillon+01)
Conclusions

• K-means Clustering
  ○ Convergence modelled by dynamical system.
  ○ Helped by seeding w/ deterministic method.

• Performance of fast SVM via clustering.
  ○ Speeded up in practice
  ○ Proved theoretical bound.
See poster for details.

• Low Memory Factored Representation.
  ○ Cluster w/out computing pairwise distances.
  ○ Compact representation, easily updatable.
  ○ Ideally, would like clustering to be faster than linear.
  ○ Easily used for various applications: clustering, IR, streaming.