COMS 4772 Fall 2015

Lecture 1
Selected topics in machine learning theory

• Mostly topics related to “unsupervised learning”.

• A tentative list (subject to time constraints):
  • Dimensionality reduction
  • Principal component analysis
  • Clustering
  • Sparse coding
  • Maximum entropy modeling
  • Online optimization and boosting

• We’ll look at algorithmic techniques for these tasks, and mathematical tools for analyzing their behavior.
Example: clustering
What is clustering for?

1. One use of clustering is to find “meaningful groups” in data.  
   • Example: your data comes from a mixture of $k$ subpopulations, and you want to learn about each of these subpopulations.

2. A different use is simply to find a small, finite approximation for very large or even infinite set.  
   • Pick an objective function that measures how good the approximation is, and try to optimize it.

   Let’s consider this second use.
**k-means clustering**

- Given a data set $S \subseteq \mathbb{R}^d$, pick $k$ points $T = \{y_1, y_2, \ldots, y_k\} \subset \mathbb{R}^d$ so as to minimize $\text{cost}(T) = \sum_{x \in S} \min_{y \in T} \|x - y\|_2^2$.

- Common question: “How do I pick $k$”?

- Cost of optimal $k$-means clustering decreases with $k$, perhaps like 

  $O\left(\frac{1}{\sqrt{k}}\right)$ or $O\left(\frac{1}{k}\right)$.

- But $k$-means clustering is NP-hard, so we can’t always compute the optimal solution efficiently.
Approximate $k$-means clustering

• A $c$-approximation algorithm returns a solution $T_k$ with cost
  \[ \text{cost}(T_k) \leq c \cdot \text{cost}(\text{OPT}_k) \]

• For example, $k$-means++ is an $O(\log k)$-approximation algorithm.
  
  • If $\text{cost}(\text{OPT}_k) = O \left( \frac{1}{\sqrt{k}} \right)$, then $\text{cost}(T_k) = O \left( \frac{\log k}{\sqrt{k}} \right)$.

• Actually, $k$-means++ is also an $(a, b)$-approximation algorithm for some constants $a, b > 0$: returns a solution $T_k$ with cost
  \[ \text{cost}(T_k) \leq b \cdot \text{cost}(\text{OPT}_{k/a}) \]

  • If $\text{cost}(\text{OPT}_k) = O \left( \frac{1}{\sqrt{k}} \right)$, then $\text{cost}(T_k) = O \left( \frac{b}{\sqrt{k/a}} \right) = O \left( \frac{1}{\sqrt{k/a}} \right)$. 
Mixture models

Now consider using clustering to find meaningful groups:

• Mixture model: data comes from a mixture of $k$ distinct distributions.
• **Goal**: learn about the individual distributions in the mixture.
• Can a $c$-approximate $k$-means algorithm help?
• Can dimensionality reduction help?
Example: latent semantic analysis
Representing a corpus of documents

• Represent a corpus of documents by counts of words they contain:

\[
\begin{array}{c|cccc}
 & \text{aardvark} & \text{abacus} & \text{abalone} & \cdots \\
\hline
\text{document 1} & 3 & 0 & 0 & \cdots \\
\text{document 2} & 7 & 0 & 4 & \cdots \\
\text{document 3} & 2 & 4 & 0 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\end{array}
\]

• Matrix \( A \in \mathbb{R}^{n \times d} \), one row per document, one column per word.
• \( A_{i,j} = \# \) times word \( j \) appears in document \( i \).
Computing similarity between documents

- **Given a new document, how similar is it to existing documents?**
- **Measure of similarity:** (normalized) inner product between word count vectors.
- **Naïvely takes** $O(nd)$ **time to compute similarity between new document and existing documents.**
  
  [ Or at least $O(\text{nnz}(A))$ time. ]
- **However, if $A$ is approximately low-rank, we could speed this up.**
- **Why could we hope for a good low-rank approximation of $A$?**
- **How can we compute such an approximation?**
Low-rank structure in latent semantic analysis

• A plausible modeling assumption:
  • $k \ll \min\{n, d\}$ topics, each represented by a distribution over words:
    $$\beta_1, \beta_2, \ldots, \beta_k \in \mathbb{R}^d$$
    (each $\beta_i$ is a probability vector).
  • Each document $i$ is associated with a mixture of the $k$ topics
    (i.e., a distribution over the $k$ topics).
  • Document $i$’s count vector is drawn from a multinomial distribution with
    probabilities given by the document’s mixture of the $\beta_i$’s (“bag of words”).

• This implies that in expectation, the matrix $A$ has rank $\ll \min\{n, d\}$. 
Finding a low-rank approximation

- Under certain objective measures, optimal low-rank approximation is given by **low-rank singular value decomposition (SVD)**.
- How fast can we compute the SVD? $O(nd^2)$ or $O(n^2d)$ time.
  - Slow when $n$ and $d$ are large.
  - BUT: this is wasteful because we only want the top few components anyway.
- Can we compute a low-rank approximation more quickly?
Approximating a low-rank approximation

• “Sketch-and-solve” algorithm:
  • Apply a random linear map to every column of \( A \), call the resulting matrix \( B \).
    • For example, a random partial Fourier transform.
  • Use a fast least squares algorithm to approximately project the rows of \( A \) onto the row space of \( B \).
    • Such a fast least squares algorithm can be devised using random linear maps.
  • Compute a low-rank approximation of the projected rows of \( A \).

• Running time is \( \tilde{O}\left( (n + d)k^2 + \text{poly}(k) \right) \) for rank-\( k \) approximation.
  • Compare to \( O(nd^2) \) or \( O(n^2d) \) time for full SVD.

• Output is a near-optimal low-rank approximation to \( A \).
More about the course
What this course is not

• A course on the theory of supervised learning.
  • See COMS 4254 (Introduction to Computational Learning Theory).

• A course on probabilistic modeling.
  • We’ll discuss some very simple probabilistic models, but very little about their actual development or application.
  • See STAT 6509 (Foundations of Graphical Models).

• A course on how to apply PCA, $k$-means, etc. to your data.
  • Theory may shed some light on when these techniques are applicable.
  • But we will abstract away a lot of practical details in order to get at the mathematical core of these problems.
Course (pre-)requirements

• Prerequisites (we will assess some of these with a calibration quiz):
  • Machine learning (at the level of COMS 4771 or STAT 4400)
  • Algorithms and data structures (at the level of CSOR 4231)
  • Linear algebra and probability.
  • Also some calculus and statistics.

• Course requirements:
  • Around four problem sets (50% of the grade); see website for instructions.
  • A project and possible oral presentation (50% of the grade); details to come.
Resources

• All course information and materials will be available on the website http://www.cs.columbia.edu/~djhsu/coms4772-f15
• Instructor: Daniel Hsu
• Office hours: Wednesdays 2:30-4:30 PM in 702 CEPSR.
• Course assistants: Chang Chen, Angus Ding
• Course e-mail: coms4772@gmail.com

• More information to be posted on the website.