Ensemble Methods
+
Recommender Systems
Reminders

• Homework 9: Learning Paradigms
  – Out: Wed, Apr 24
  – Due: Wed, May 1 at 11:59pm
  – Can only be submitted up to 3 days late, so we can return grades before final exam

• Today’s In-Class Poll
### Q&A

**Q:** In k-Means, since we don’t have a validation set, how do we pick \( k \)?

**A:** Look at the training objective function as a function of \( k \) \( J(c, z) \) and pick the value at the “elbo” of the curve.

**Q:** What if our random initialization for k-Means gives us poor performance?

**A:** Do **random restarts**: that is, run k-means from scratch, say, 10 times and pick the run that gives the lowest training objective function value. The objective function is **nonconvex**, so we’re just looking for the best local minimum.
Learning Paradigms:
What data is available and when? What form of prediction?
- supervised learning
- unsupervised learning
- semi-supervised learning
- reinforcement learning
- active learning
- imitation learning
- domain adaptation
- online learning
- density estimation
- recommender systems
- feature learning
- manifold learning
- dimensionality reduction
- ensemble learning
- distant supervision
- hyperparameter optimization

Problem Formulation:
What is the structure of our output prediction?
- boolean: Binary Classification
- categorical: Multiclass Classification
- ordinal: Ordinal Classification
- real: Regression
- ordering: Ranking
- multiple discrete: Structured Prediction
- multiple continuous: (e.g. dynamical systems)
- both discrete & cont.: (e.g. mixed graphical models)

Facets of Building ML Systems:
How to build systems that are robust, efficient, adaptive, effective?
1. Data prep
2. Model selection
3. Training (optimization / search)
4. Hyperparameter tuning on validation data
5. (Blind) Assessment on test data

Theoretical Foundations:
What principles guide learning?
- probabilistic
- information theoretic
- evolutionary search
- ML as optimization

Big Ideas in ML:
Which are the ideas driving development of the field?
- inductive bias
- generalization / overfitting
- bias-variance decomposition
- generative vs. discriminative
- deep nets, graphical models
- PAC learning
- distant rewards

Application Areas:
Key challenges? NLP, Speech, Computer Vision, Robotics, Medicine, Search
Outline for Today

We’ll talk about two distinct topics:

1. **Ensemble Methods**: combine or learn multiple classifiers into one (i.e. a family of algorithms)

2. **Recommender Systems**: produce recommendations of what a user will like (i.e. the solution to a particular type of task)

We’ll use a prominent example of a recommender systems (the Netflix Prize) to motivate both topics...
RECOMMENDER SYSTEMS
Recommender Systems

A Common Challenge:

– Assume you’re a company selling *items* of some sort: movies, songs, products, etc.

– Company collects millions of *ratings* from *users* of their *items*

– To maximize profit / user happiness, you want to *recommend* items that users are likely to want
Recommender Systems
Recommender Systems
Recommender Systems

The Netflix Prize sought to substantially improve the accuracy of predictions about how much someone is going to enjoy a movie based on their movie preferences.

On September 21, 2009 we awarded the $1M Grand Prize to team “BellKor’s Pragmatic Chaos”. Read about their algorithm, checkout team scores on the Leaderboard, and join the discussions on the Forum.

We applaud all the contributors to this quest, which improves our ability to connect people to the movies they love.
Recommender Systems

Problem Setup

- 500,000 users
- 20,000 movies
- 100 million ratings
- Goal: To obtain lower root mean squared error (RMSE) than Netflix’s existing system on 3 million held out ratings
ENSEMBLE METHODS
Recommender Systems

Top performing systems were ensembles
Weighted Majority Algorithm

(Littlestone & Warmuth, 1994)

- **Given**: pool $A$ of binary classifiers (that you know nothing about)
- **Data**: stream of examples (i.e. online learning setting)
- **Goal**: design a new learner that uses the predictions of the pool to make new predictions
- **Algorithm**:
  - Initially weight all classifiers equally
  - Receive a training example and predict the (weighted) majority vote of the classifiers in the pool
  - Down-weight classifiers that contribute to a mistake by a factor of $\beta$
Weighted Majority Algorithm
(Littlestone & Warmuth, 1994)

Suppose we have a pool of $T$ binary classifiers $\mathcal{A} = \{h_1, \ldots, h_T\}$ where $h_t : \mathbb{R}^M \rightarrow \{+1, -1\}$. Let $\alpha_t$ be the weight for classifier $h_t$.

**Algorithm 1 Weighted Majority Algorithm**

1: procedure WEIGHTEDMAJORITY($\mathcal{A}, \beta$)
2: Initialize classifier weights $\alpha_t = 1, \forall t \in \{1, \ldots, T\}$
3: for each training example $(x, y)$ do
4: Predict majority vote class (splitting ties randomly)

$$\hat{h}(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$$

5: if a mistake is made $\hat{h}(x) \neq y$ then
6: for each classifier $t \in \{1, \ldots, T\}$ do
7: If $h_t(x) \neq y$, then $\alpha_t \leftarrow \beta \alpha_t$
Weighted Majority Algorithm
(Littlestone & Warmuth, 1994)

**Theorem 0.1** (Littlestone & Warmuth, 1994). *If the Weighted Majority Algorithm is applied to a pool $\mathcal{A}$ of classifiers, and if each algorithm makes at most $m$ mistakes on the sequence of examples, then the total number of mistakes is upper bounded by $2.4(\log |\mathcal{A}| + m)$."

This is a “mistake bound” of the variety we saw for the Perceptron algorithm.
ADABOOST
Comparison

Weighted Majority Algorithm
• an example of an ensemble method
• assumes the classifiers are learned ahead of time
• only learns (majority vote) weight for each classifiers

AdaBoost
• an example of a boosting method
• simultaneously learns:
  – the classifiers themselves
  – (majority vote) weight for each classifiers
AdaBoost: Toy Example

$D_1$

weak classifiers = vertical or horizontal half-planes
AdaBoost: Toy Example

$h_1$

$\varepsilon_1 = 0.30$

$\alpha_1 = 0.42$

$D_2$

Slide from Schapire NIPS Tutorial
AdaBoost: Toy Example

\[ \alpha_2 = 0.65 \]

\[ \epsilon_2 = 0.21 \]
AdaBoost: Toy Example

Slide from Schapire NIPS Tutorial
AdaBoost: Toy Example

\[ H_{\text{final}} = \text{sign} \left( 0.42 + 0.65 + 0.92 \right) \]

Slide from Schapire NIPS Tutorial
AdaBoost

Given: \((x_1, y_1), \ldots, (x_m, y_m)\) where \(x_i \in X, y_i \in Y = \{-1, +1\}\)
Initialize \(D_1(i) = 1/m\).
For \(t = 1, \ldots, T\):

- Train weak learner using distribution \(D_t\).
- Get weak hypothesis \(h_t : X \rightarrow \{-1, +1\}\) with error
  \[
  \epsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i].
  \]
- Choose \(\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)\).
- Update:
  \[
  D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} 
    e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\
    e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i 
  \end{cases}
  = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}
  \]
  where \(Z_t\) is a normalization factor (chosen so that \(D_{t+1}\) will be a distribution).

Output the final hypothesis:

\[
H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right).
\]
Figure 2: Error curves and the margin distribution graph for boosting C4.5 on the letter dataset as reported by Schapire et al. [41]. *Left:* the training and test error curves (lower and upper curves, respectively) of the combined classifier as a function of the number of rounds of boosting. The horizontal lines indicate the test error rate of the base classifier as well as the test error of the final combined classifier. *Right:* The cumulative distribution of margins of the training examples after 5, 100 and 1000 iterations, indicated by short-dashed, long-dashed (mostly hidden) and solid curves, respectively.
Learning Objectives

Ensemble Methods / Boosting

You should be able to...

1. Implement the Weighted Majority Algorithm
2. Implement AdaBoost
3. Distinguish what is learned in the Weighted Majority Algorithm vs. Adaboost
4. Contrast the theoretical result for the Weighted Majority Algorithm to that of Perceptron
5. Explain a surprisingly common empirical result regarding Adaboost train/test curves
Outline

• **Recommender Systems**
  – Content Filtering
  – Collaborative Filtering (CF)
  – CF: Neighborhood Methods
  – CF: Latent Factor Methods

• **Matrix Factorization**
  – Background: Low-rank Factorizations
  – Residual matrix
  – Unconstrained Matrix Factorization
    • Optimization problem
    • Gradient Descent, SGD, Alternating Least Squares
    • User/item bias terms (matrix trick)
  – Singular Value Decomposition (SVD)
  – Non-negative Matrix Factorization
RECOMMENDER SYSTEMS
Recommender Systems

Problem Setup

• 500,000 users
• 20,000 movies
• 100 million ratings
• Goal: To obtain lower root mean squared error (RMSE) than Netflix’s existing system on 3 million held out ratings
Recommender Systems

![Leaderboard](image)

**Leaderboard**

Showing Test Score. Click here to show quiz score

<table>
<thead>
<tr>
<th>Rank</th>
<th>Team Name</th>
<th>Best Test Score</th>
<th>% Improvement</th>
<th>Best Submit Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BellKor's Pragmatic Chaos</td>
<td>0.8567</td>
<td>10.06</td>
<td>2009-07-26 18:18:28</td>
</tr>
<tr>
<td>2</td>
<td>The Ensemble</td>
<td>0.8567</td>
<td>10.06</td>
<td>2009-07-26 18:38:22</td>
</tr>
<tr>
<td>3</td>
<td>Grand Prize Team</td>
<td>0.8562</td>
<td>9.90</td>
<td>2009-07-10 21:24:40</td>
</tr>
<tr>
<td>4</td>
<td>Opera Solutions and Vandelay United</td>
<td>0.8588</td>
<td>9.84</td>
<td>2009-07-10 01:12:31</td>
</tr>
<tr>
<td>5</td>
<td>Vandelay Industries II</td>
<td>0.8591</td>
<td>9.81</td>
<td>2009-07-10 00:32:20</td>
</tr>
<tr>
<td>6</td>
<td>PragmaticTheory</td>
<td>0.8594</td>
<td>9.77</td>
<td>2009-06-24 12:06:56</td>
</tr>
<tr>
<td>7</td>
<td>BellKor in BigChaos</td>
<td>0.8601</td>
<td>9.70</td>
<td>2009-05-13 08:14:09</td>
</tr>
<tr>
<td>8</td>
<td>Dace</td>
<td>0.8612</td>
<td>9.59</td>
<td>2009-07-24 17:18:43</td>
</tr>
<tr>
<td>9</td>
<td>Feeds2</td>
<td>0.8622</td>
<td>9.48</td>
<td>2009-07-12 13:11:51</td>
</tr>
<tr>
<td>10</td>
<td>BigChaos</td>
<td>0.8623</td>
<td>9.47</td>
<td>2009-04-07 12:33:59</td>
</tr>
<tr>
<td>11</td>
<td>Opera Solutions</td>
<td>0.8623</td>
<td>9.47</td>
<td>2009-07-24 00:34:07</td>
</tr>
<tr>
<td>12</td>
<td>BellKor</td>
<td>0.8624</td>
<td>9.46</td>
<td>2009-07-26 17:19:11</td>
</tr>
</tbody>
</table>
Recommender Systems

• **Setup:**
  - **Items:** movies, songs, products, etc. (often many thousands)
  - **Users:** watchers, listeners, purchasers, etc. (often many millions)
  - **Feedback:** 5-star ratings, not-clicking ‘next’, purchases, etc.

• **Key Assumptions:**
  - Can represent ratings numerically as a user/item matrix
  - Users only rate a small number of items (the matrix is sparse)

<table>
<thead>
<tr>
<th></th>
<th>Doctor</th>
<th>Strange</th>
<th>Star Trek: Beyond</th>
<th>Zootopia</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Charlie</td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Two Types of Recommender Systems

Content Filtering

- **Example:** Pandora.com music recommendations (Music Genome Project)
- **Con:** Assumes access to side information about items (e.g. properties of a song)
- **Pro:** Got a new item to add? No problem, just be sure to include the side information

Collaborative Filtering

- **Example:** Netflix movie recommendations
- **Pro:** Does not assume access to side information about items (e.g. does not need to know about movie genres)
- **Con:** Does not work on new items that have no ratings
COLLABORATIVE FILTERING
Collaborative Filtering

• Everyday Examples of Collaborative Filtering...
  – Bestseller lists
  – Top 40 music lists
  – The “recent returns” shelf at the library
  – Unmarked but well-used paths thru the woods
  – The printer room at work
  – “Read any good books lately?”
  – …

• Common insight: personal tastes are correlated
  – If Alice and Bob both like X and Alice likes Y then Bob is more likely to like Y
  – especially (perhaps) if Bob knows Alice
Two Types of Collaborative Filtering

1. Neighborhood Methods

2. Latent Factor Methods

Figures from Koren et al. (2009)
Two Types of Collaborative Filtering

1. Neighborhood Methods

In the figure, assume that a green line indicates the movie was watched.

Algorithm:
1. **Find neighbors** based on similarity of movie preferences
2. **Recommend** movies that those neighbors watched

Figures from Koren et al. (2009)
Two Types of Collaborative Filtering

2. Latent Factor Methods

- Assume that both movies and users live in some low-dimensional space describing their properties
- **Recommend** a movie based on its **proximity** to the user in the latent space
- **Example Algorithm:** Matrix Factorization

Figures from Koren et al. (2009)
MATRIX FACTORIZATION
Recommending Movies

**Question:**
Applied to the Netflix Prize problem, which of the following methods always requires side information about the users and movies?

*Select all that apply*
A. collaborative filtering  
B. latent factor methods  
C. ensemble methods  
D. content filtering  
E. neighborhood methods  
F. recommender systems

**Answer:**

Matrix Factorization

• Many different ways of factorizing a matrix
• We’ll consider three:
  1. Unconstrained Matrix Factorization
  2. Singular Value Decomposition
  3. Non-negative Matrix Factorization

• MF is just another example of a common recipe:
  1. define a model
  2. define an objective function
  3. optimize with SGD
Matrix Factorization

Whiteboard

– Background: Low-rank Factorizations
– Residual matrix
Example: MF for Netflix Problem

(a) Example of rank-2 matrix factorization

(b) Residual matrix

Figures from Aggarwal (2016)
Regression vs. Collaborative Filtering

The figure illustrates the differences between regression and collaborative filtering.

**Regression**
- **Independent Variables**: Represented by the horizontal axis.
- **Dependent Variable**: Represented by the vertical axis.
- **Training Rows**: Shaded entries are missing and need to be predicted.
- **Test Rows**: The value of the missing entries need to be learned for the test data.

**Collaborative Filtering**
- **Independent Variables**: Represented by the horizontal axis.
- **Dependent Variable**: Represented by the vertical axis.
- **Training Rows**: Shaded entries are missing and need to be predicted.
- **Test Rows**: The value of the missing entries need to be learned for the test data.
- **No Demarcation Between Dependent and Independent Variables**
- **No Demarcation Between Training and Test Rows**

*Figures from Aggarwal (2016)*
UNCONSTRAINED MATRIX FACTORIZATION
Unconstrained Matrix Factorization

Whiteboard

– Optimization problem
– SGD
– SGD with Regularization
– Alternating Least Squares
– User/item bias terms (matrix trick)
Unconstrained Matrix Factorization

In-Class Exercise

Derive a block coordinate descent algorithm for the Unconstrained Matrix Factorization problem.

- **User vectors:** 
  \[ \mathbf{w}_u \in \mathbb{R}^r \]

- **Item vectors:** 
  \[ \mathbf{h}_i \in \mathbb{R}^r \]

- **Rating prediction:** 
  \[ v_{ui} = \mathbf{w}_u^T \mathbf{h}_i \]

- **Set of non-missing entries** 
  \[ \mathcal{Z} = \{(u, i) : v_{ui} \text{ is observed}\} \]

- **Objective:** 
  \[ \arg\min_{\mathbf{w}, \mathbf{h}} \sum_{(u, i) \in \mathcal{Z}} (v_{ui} - \mathbf{w}_u^T \mathbf{h}_i)^2 \]
Matrix Factorization
(with matrices)

- **User vectors:**
  \[(W_{u*})^T \in \mathbb{R}^r\]
- **Item vectors:**
  \[H_{*i} \in \mathbb{R}^r\]
- **Rating prediction:**
  \[V_{ui} = W_{u*}H_{*i} = [WH]_{ui}\]
Matrix Factorization (with vectors)

- **User vectors:**
  \[ w_u \in \mathbb{R}^r \]

- **Item vectors:**
  \[ h_i \in \mathbb{R}^r \]

- **Rating prediction:**
  \[ v_{ui} = w_u^T h_i \]

Figures from Koren et al. (2009)
Matrix Factorization
(with vectors)

- Set of non-missing entries:
  \[ \mathcal{Z} = \{(u, i) : v_{ui} \text{ is observed}\} \]

- Objective:
  \[
  \arg\min_{w,h} \sum_{(u,i) \in \mathcal{Z}} (v_{ui} - w_u^T h_i)^2
  \]

Figures from Koren et al. (2009)
Matrix Factorization (with vectors)

- Regularized Objective:

\[
\arg\min_{w, h} \sum_{(u, i) \in \mathcal{Z}} (v_{ui} - w_u^T h_i)^2 \\
+ \lambda \left( \sum_i \|w_i\|^2 + \sum_u \|h_u\|^2 \right)
\]
Matrix Factorization (with vectors)

• Regularized Objective:

$$\text{argmin}_{w,h} \sum_{(u,i) \in \mathcal{Z}} (v_{ui} - w_u^T h_i)^2$$

$$+ \lambda \left( \sum_i \|w_i\|^2 + \sum_u \|h_u\|^2 \right)$$

• SGD update for random (u,i):

$$e_{ui} \leftarrow v_{ui} - w_u^T h_i$$

$$w_u \leftarrow w_u + \gamma (e_{ui} h_i - \lambda w_u)$$

$$h_i \leftarrow h_i + \gamma (e_{ui} w_u - \lambda h_i)$$
Matrix Factorization (with matrices)

- **User vectors:**
  \[(W_u)^T \in \mathbb{R}^r\]

- **Item vectors:**
  \[H_i \in \mathbb{R}^r\]

- **Rating prediction:**
  \[V_{ui} = W_u^* H_i \]
  \[= [WH]_{ui}\]
Matrix Factorization (with matrices)

- SGD

require that the loss can be written as

$$L = \sum_{(i,j) \in Z} l(V_{ij}, W_{i*}, H_{*j})$$

Algorithm 1 SGD for Matrix Factorization

Require: A training set $Z$, initial values $W_0$ and $H_0$

while not converged do {step}

Select a training point $(i, j) \in Z$ uniformly at random.

$W'_{i*} \leftarrow W_{i*} - \epsilon_n N \frac{\partial}{\partial W_{i*}} l(V_{ij}, W_{i*}, H_{*j})$

$H_{*j} \leftarrow H_{*j} - \epsilon_n N \frac{\partial}{\partial H_{*j}} l(V_{ij}, W_{i*}, H_{*j})$

$W_{i*} \leftarrow W'_{i*}$

end while
Figure 3. The first two vectors from a matrix decomposition of the Netflix Prize data. Selected movies are placed at the appropriate spot based on their factor vectors in two dimensions. The plot reveals distinct genres, including clusters of movies with strong female leads, fraternity humor, and quirky independent films.

Figure from Koren et al. (2009)
Matrix Factorization

Comparison of Optimization Algorithms

ALS = alternating least squares

Figure from Gemulla et al. (2011)
SVD FOR COLLABORATIVE FILTERING
Singular Value Decomposition for Collaborative Filtering

For any arbitrary matrix $A$, SVD gives a decomposition:

$$A = UV^T$$

where $\Lambda$ is a diagonal matrix, and $U$ and $V$ are orthogonal matrices.

Suppose we have the SVD of our ratings matrix

$$R = Q\Sigma P^T,$$

but then we truncate each of $Q$, $\Sigma$, and $P$ s.t. $Q$ and $P$ have only $k$ columns and $\Sigma$ is $k \times k$:

$$R \approx Q_k \Sigma_k P_k^T$$

For collaborative filtering, let:

$$U \triangleq Q_k \Sigma_k$$
$$V \triangleq P_k$$

$$\Rightarrow U, V = \arg\min_{U,V} \frac{1}{2} \| R - UV^T \|_2^2$$

s.t. columns of $U$ are mutually orthogonal
s.t. columns of $V$ are mutually orthogonal

**Theorem:** If $R$ fully observed and no regularization, the optimal $UV^T$ from SVD equals the optimal $UV^T$ from Unconstrained MF
NON-NEGATIVE MATRIX FACTORIZATION
Implicit Feedback Datasets

• What information does a five-star rating contain?

• Implicit Feedback Datasets:
  – In many settings, users don’t have a way of expressing dislike for an item (e.g. can’t provide negative ratings)
  – The only mechanism for feedback is to “like” something

• Examples:
  – Facebook has a “Like” button, but no “Dislike” button
  – Google’s “+1” button
  – Pinterest pins
  – Purchasing an item on Amazon indicates a preference for it, but there are many reasons you might not purchase an item (besides dislike)
  – Search engines collect click data but don’t have a clear mechanism for observing dislike of a webpage

Examples from Aggarwal (2016)
Non-negative Matrix Factorization

Constrained Optimization Problem:

\[ U, V = \arg\min_{U,V} \frac{1}{2} \| R - UV^T \|_2^2 \]

s.t. \( U_{ij} \geq 0 \)

s.t. \( V_{ij} \geq 0 \)

**Multiplicative Updates**: simple iterative algorithm for solving just involves multiplying a few entries together
Summary

• Recommender systems solve many real-world (*large-scale) problems

• Collaborative filtering by Matrix Factorization (MF) is an efficient and effective approach

• MF is just another example of a common recipe:
  1. define a model
  2. define an objective function
  3. optimize with SGD