CSE 158
Web Mining and Recommender Systems

Midterm recap
Midterm on Wednesday!

- 5:10 pm – 6:10 pm
- Closed book – but I’ll provide a similar level of basic info as in the last page of previous midterms
CSE 158
Web Mining and Recommender Systems

Week 1 recap
Learning approaches attempt to model data in order to solve a problem.

Unsupervised learning approaches find patterns/relationships/structure in data, but are not optimized to solve a particular predictive task.
- E.g. PCA, community detection

Supervised learning aims to directly model the relationship between input and output variables, so that the output variables can be predicted accurately given the input.
- E.g. linear regression, logistic regression
Linear regression assumes a predictor of the form

\[ X \theta = y \]

(or \( A x = b \) if you prefer)

matrix of features (data)

unknowns (which features are relevant)

vector of outputs (labels)
Regression diagnostics

**Mean-squared error (MSE)**

\[
\frac{1}{N} \| y - X \theta \|_2^2
\]

\[
= \frac{1}{N} \sum_{i=1}^{N} (y_i - X_i \cdot \theta)^2
\]
Representing the month as a feature

How would you build a feature to represent the **month**?

Nov 4, 2019, Monday

\[
\begin{bmatrix}
0 & \ldots & 1 & \ldots & 0 \\
00 & \ldots & 1 & \ldots & 00 \\
01 & 6 & 00 & 00
\end{bmatrix}
\]

\[\text{Nov} \quad 4 \quad \text{Monday}\]
Representing the month as a feature
Occam’s razor

“Among competing hypotheses, the one with the fewest assumptions should be selected”
Regularization is the process of penalizing model complexity during training.

\[ \text{arg min}_\theta = \frac{1}{N} \| y - X \theta \|^2_2 + \lambda \| \theta \|^2_2 \]

How much should we trade-off accuracy versus complexity?
A validation set is constructed to “tune” the model’s parameters

- Training set: used to optimize the model’s parameters
- Test set: used to report how well we expect the model to perform on unseen data
- Validation set: used to tune any model parameters that are not directly optimized

\[ \text{choose} \]

\( \text{Only use once!} \)
Regularization
Model selection

A few “theorems” about training, validation, and test sets

- The training error **increases** as lambda **increases**
- The validation and test error are at least as large as the training error (assuming infinitely large random partitions)
- The validation/test error will usually have a “sweet spot” between under- and over-fitting
Classification

Will I purchase this product?
(Yes)

Will I click on this ad?
(No)
What animal appears in this image?

(mandarin duck)
What are the *categories* of the item being described?
(book, fiction, philosophical fiction)

From Booklist

Houellebecq's deeply philosophical novel is about an alienated young man searching for happiness in the computer age. Bored with the world and too weary to try to adapt to the foibles of friends and coworkers, he retreats into himself, descending into depression while attempting to analyze the passions of the people around him. Houellebecq uses his nameless narrator as a vehicle for extended exploration into the meanings and manifestations of love and desire in human interactions. Ironically, as the narrator attempts to define love in increasingly abstract terms, he becomes less and less capable of experiencing that which he is so desperate to understand. Intelligent and well written, the short novel is a thought-provoking inspection of a generation's confusion about all things sexual. Houellebecq captures precisely the cynical disillusionment of disaffected youth. Bonnie Johnston --This text refers to an out of print or unavailable edition of this title.
Linear regression assumes a predictor of the form

\[ X\theta = y \]

- Matrix of features (data)
- Unknowns (which features are relevant)
- Vector of outputs (labels)
Regression vs. classification

But how can we predict **binary** or **categorical** variables?

\[ f(\text{data}) \rightarrow \text{labels} \]

\{0,1\}, \{True, False\}

\{1, \ldots, N\}
We’ll attempt to build classifiers that make decisions according to rules of the form

\[ y_i = \begin{cases} 
1 & \text{if } X_i \cdot \theta > 0 \\
0 & \text{otherwise}
\end{cases} \]
In week 2

1. Naïve Bayes
Assumes an independence relationship between the features and the class label and “learns” a simple model by counting

2. Logistic regression
Adapts the regression approaches we saw last week to binary problems

3. Support Vector Machines
Learns to classify items by finding a hyperplane that separates them
Naïve Bayes (2 slide summary)

\[
(f_{\text{feature}_i} \perp f_{\text{feature}_j} | \text{label})
\]

\[
p(f_{\text{feature}_i}, f_{\text{feature}_j} | \text{label})
\]

\[
= p(f_{\text{feature}_i} | \text{label})p(f_{\text{feature}_j} | \text{label})
\]
Naïve Bayes (2 slide summary)

\[ p(y|\text{feat}) = \frac{p(y) \prod p(f_i|y)}{p(\text{feat})} \]
Double-counting: naïve Bayes vs Logistic Regression

Q: What would happen if we trained two regressors, and attempted to “naively” combine their parameters?

\[
\text{no. of pages} = \alpha + \beta_1 \cdot \delta(\text{mentions wizards})
\]

\[
\text{no. of pages} = \alpha + \beta_2 \cdot \delta(\text{mentions witches})
\]

\[
\text{no. of pages} = \alpha + \beta_1 \cdot \delta(\text{mentions wizards}) + \beta_2 \cdot \delta(\text{mentions witches})
\]
Logistic regression

**sigmoid function:**  \( \sigma(t) = \frac{1}{1+e^{-t}} \)
Logistic regression

**Training:**

\[ X_i \cdot \theta \] should be maximized when \( y_i \) is positive and minimized when \( y_i \) is negative

\[
\arg \max_{\theta} \prod_i \delta(y_i = 1) p_{\theta}(y_i | X_i) + \delta(y_i = 0) (1 - p_{\theta}(y_i | X_i))
\]

\( \delta(\text{arg}) = 1 \) if the argument is true, \( = 0 \) otherwise
Logistic regression

\[ \arg \max_{\theta} \prod_i \delta(y_i = 1)p_\theta(y_i | X_i) + \delta(y_i = 0)(1 - p_\theta(y_i | X_i)) \]
Q: Where would a logistic regressor place the decision boundary for these features?

- Easy to classify: positive examples
- Hard to classify: negative examples

- Easy to classify: positive examples
- Hard to classify: negative examples
Logistic regression

- Logistic regressors don’t optimize the number of “mistakes”
- No special attention is paid to the “difficult” instances – every instance influences the model
- But “easy” instances can affect the model (and in a bad way!)
- How can we develop a classifier that optimizes the number of mislabeled examples?
Support Vector Machines

\[ \theta x - \alpha = 1 \]
\[ \theta x - \alpha = 0 \]
\[ \theta x - \alpha = -1 \]

\[ \text{arg min}_{\theta, \alpha} \frac{1}{2} \| \theta \|_2^2 \]

such that
\[ \forall i y_i (\theta \cdot X_i - \alpha) \geq 1 \]

“support vectors”
The classifiers we’ve seen in Week 2 all attempt to make decisions by associating weights (theta) with features (x) and classifying according to

\[ y_i = \begin{cases} 
1 & \text{if } X_i \cdot \theta > 0 \\
0 & \text{otherwise}
\end{cases} \]
Summary

• **Naïve Bayes**
  • Probabilistic model (fits $p(label|data)$)
  • Makes a conditional independence assumption of the form $(feature_i \perp \!\!\!\!\!\perp feature_j|label)$ allowing us to define the model by computing $p(feature_i|label)$ for each feature
  • Simple to compute just by counting

• **Logistic Regression**
  • Fixes the “double counting” problem present in naïve Bayes

• **SVMs**
  • Non-probabilistic: optimizes the classification error rather than the likelihood
Which classifier is best?

1. When data are highly imbalanced

   If there are far fewer positive examples than negative examples we may want to assign additional weight to negative instances (or vice versa)

   e.g. will I purchase a product? If I purchase 0.00001% of products, then a classifier which just predicts “no” everywhere is 99.99999% accurate, but not very useful
Which classifier is best?

2. When mistakes are more costly in one direction
   False positives are nuisances but false negatives are disastrous (or vice versa)

  e.g. which of these bags contains a weapon?
3. When we only care about the “most confident” predictions

E.g. does a relevant result appear among the first page of results?
Evaluating classifiers

decision boundary

negative  positive
Evaluating classifiers

<table>
<thead>
<tr>
<th>Label</th>
<th>true</th>
<th>false</th>
</tr>
</thead>
<tbody>
<tr>
<td>true positive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>false positive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>false negative</td>
<td></td>
<td>true negative</td>
</tr>
</tbody>
</table>

Classification accuracy
= correct predictions / #predictions
= (TP + TN) / (TP + TN + FP + FN)

Error rate
= incorrect predictions / #predictions
= (FP + FN) / (TP + TN + FP + FN)
• Linear classification – know what the different classifiers are and when you should use each of them. What are the advantages/disadvantages of each
• Know how to evaluate classifiers – what should you do when you care more about false positives than false negatives etc.
CSE 158 – Lecture 10
Web Mining and Recommender Systems

Week 3
Why dimensionality reduction?

Goal: take **high-dimensional** data, and describe it compactly using a small number of dimensions.

Assumption: Data lies (approximately) on some **low-dimensional** manifold (a few dimensions of opinions, a small number of topics, or a small number of communities).
Principal Component Analysis

\[ X \]

\[ Y \]

\[ x_1 \]
\[ x_0 \]

\[ \phi \]
rotate

\[ \phi^T \]
un-rotate

discard lowest-variance dimensions

\[ Y, 0 \]
Principal Component Analysis

Construct such vectors from 100,000 patches from real images and run PCA:

Color:
Principal Component Analysis

• We want to find a low-dimensional representation that best compresses or “summarizes” our data
• To do this we’d like to keep the dimensions with the highest variance (we proved this), and discard dimensions with lower variance. Essentially we’d like to capture the aspects of the data that are “hardest” to predict, while discard the parts that are “easy” to predict
• This can be done by taking the eigenvectors of the covariance matrix (we didn’t prove this, but it’s right there in the slides)
Q: What would PCA do with this data?
A: Not much, variance is about equal in all dimensions
But: The data are highly clustered

Idea: can we compactly describe the data in terms of cluster memberships?
K-means Clustering

1. Input is still a matrix of features:

\[
X = \begin{pmatrix}
5 & 3 & \cdots & 1 \\
4 & 2 & & 1 \\
3 & 1 & & 3 \\
2 & 2 & & 4 \\
1 & 5 & & 2 \\
\vdots & \vdots & \ddots & \vdots \\
1 & 2 & \cdots & 1 \\
\end{pmatrix}
\]

2. Output is a list of cluster “centroids”:

\[
\text{centroids} = \begin{pmatrix}
1.1 & 2.1 \\
3.5 & 1.8 \\
0.2 & 0.1 \\
3.0 & -0.3 \\
\end{pmatrix}
\]

3. From this we can describe each point in X by its cluster membership:

\[
f = [0,0,1,0] \quad f = [0,0,0,1]
\]

\[
Y = (1, 2, 4, 3, 4, 2, 4, 2, 2, 3, 3, 2, 1, 1, 3, \ldots, 2)
\]
1. Initialize C (e.g. at random)
2. Do
3. Assign each $X_i$ to its nearest centroid
4. Update each centroid to be the mean of points assigned to it
5. While (assignments change between iterations)

(also: reinitialize clusters at random should they become empty)
Q: What if our clusters are **hierarchical**?

A: We’d like a representation that encodes that points have **some features** in common but not others
Hierarchical clustering

Hierarchical (agglomerative) clustering works by gradually fusing clusters whose points are closest together.

Assign every point to its own cluster:
Clusters = [[[1],[2],[3],[4],[5],[6],...,[N]]
While len(Clusters) > 1:
    Compute the center of each cluster
    Combine the two clusters with the nearest centers
1. Connected components

Define communities in terms of sets of nodes which are reachable from each other

- If \(a\) and \(b\) belong to a **strongly connected component** then there must be a path from \(a \rightarrow b\) and a path from \(b \rightarrow a\)
- A **weakly connected component** is a set of nodes that would be strongly connected, if the graph were undirected
What is the **Ratio Cut** cost of the following two cuts?

\[
\text{Ratio Cut}(\cdots) = \frac{1}{2} \left( \frac{3}{33} + \frac{3}{1} \right) = 1.54545
\]

\[
\text{Ratio Cut}(\cdots) = \frac{1}{2} \left( \frac{9}{16} + \frac{9}{18} \right) = 0.53125
\]
3. Clique percolation

- Clique percolation searches for “cliques” in the network of a certain size (K). Initially each of these cliques is considered to be its own community.
- If two communities share a (K-1) clique in common, they are merged into a single community.
- This process repeats until no more communities can be merged.

1. Given a clique size K
2. Initialize every K-clique as its own community
3. While (two communities I and J have a (K-1)-clique in common):
   4. Merge I and J into a single community
Week 3

- Clustering & Community detection – understand the basics of the different algorithms
  - Given some features, know when to apply PCA vs. K-means vs. hierarchical clustering
  - Given some networks, know when to apply clique percolation vs. graph cuts vs. connected components
CSE 158
Web Mining and Recommender Systems

Week 4
Definitions

Or equivalently...

\[
R = \begin{pmatrix}
1 & 0 & \cdots & 1 \\
0 & 0 & \cdots & 1 \\
\vdots & \vdots & \ddots & \vdots \\
1 & 0 & \cdots & 1
\end{pmatrix}
\]

\(R_u = \) binary representation of items purchased by \(u\)

\(R_{.,i} = \) binary representation of users who purchased \(i\)

\(I_u = \{i | R_{ui} = 1\} \quad U_i = \{u | R_{ui} = 1\}\)
Recommender Systems Concepts

• How to represent rating / purchase data as sets/matrices
• Similarity measures (Jaccard, cosine, Pearson correlation)
• Very basic ideas behind latent factor models
Jaccard similarity

\[
\text{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}
\]

\[
\text{Jaccard}(U_i, U_j) = \frac{|U_i \cap U_j|}{|U_i \cup U_j|}
\]

→ Maximum of 1 if the two users purchased **exactly the same** set of items (or if two items were purchased by the same set of users)

→ Minimum of 0 if the two users purchased **completely disjoint** sets of items (or if the two items were purchased by completely disjoint sets of users)
Cosine similarity

\[ \text{Cosine}(A, B) = \frac{A \cdot B}{\|A\| \|B\|} \]

\[ \theta = \cos^{-1} \left( \frac{A \cdot B}{\|A\| \|B\|} \right) \]

- \( \cos(\theta) = 1 \)  
  (theta = 0) \( \rightarrow \) A and B point in exactly the same direction

- \( \cos(\theta) = -1 \)  
  (theta = 180) \( \rightarrow \) A and B point in opposite directions (won’t actually happen for 0/1 vectors)

- \( \cos(\theta) = 0 \)  
  (theta = 90) \( \rightarrow \) A and B are orthogonal
Pearson correlation

Compare to the cosine similarity:

Pearson similarity (between users):

\[
\text{Sim}(u, v) = \frac{\sum_{i \in I_u \cap I_v} (R_{u,i} - \bar{R}_u)(R_{v,i} - \bar{R}_v)}{\sqrt{\sum_{i \in I_u \cap I_v} (R_{u,i} - \bar{R}_u)^2 \sum_{i \in I_u \cap I_v} (R_{v,i} - \bar{R}_v)^2}}
\]

Cosine similarity (between users):

\[
\text{Sim}(u, v) = \frac{\sum_{i \in I_u \cap I_v} R_{u,i}R_{v,i}}{\sqrt{\sum_{i \in I_u \cap I_v} R_{u,i}^2 \sum_{i \in I_u \cap I_v} R_{v,i}^2}}
\]
Rating prediction

\[ f(u, i) = \alpha + \beta_u + \beta_i \]

- \( \alpha \): how much does this user tend to rate things above the mean?
- \( \beta_u \): does this item tend to receive higher ratings than others?

**Example:**

\[ \beta_{\text{pitch black}} = -0.1 \]
\[ \beta_{\text{julian}} = -0.2 \]

\( \alpha = 4.2 \)
Latent-factor models

\[ f(u, i) = \alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i \]

- my (user’s) “preferences”
- HP’s (item) “properties”
Misc. Qs
Misc. Qs
Misc. Qs