CSE 158 Midterm Review

Exam Logistics

Time: Nov 6th;
Practice Exam: online.

Exam Topics

- Linear regression
    - Regression diagnostics
    - **Regularization**
    - **Model selection**

- Classification
    - Regression vs classification
    - Linear classification
        1. Naive Bayes
        2. Logistic regression
        3. Support vector machines (SVM)
    - Determine the best classifier
        - **Evaluating classifiers**

- Dimensionality reduction
    - Principal Component Analysis (PCA)
    - Clustering
    - **K-means Clustering -- (Greedy)**
    - **Hierarchical clustering**
        - Connected components
        - Graph cuts
        - Clique percolation

- Recommender System
    - **Jaccard similarity**
    - **Cosine similarity**
    - **Pearson Correlation**
    - Rating prediction
    - Latent-factor models

==========

Lecture 1

==========

- **Learning approaches to Model Data:**
  - **Supervised learning**: process of trying to infer from **labeled data** the underlying function that produced the labels associated with the data (**features**)
    - Infer function: f(data) → labels.
    - Ex: Movie: user and movie features as data; star rating as label.
    - Model relationships between input and output.
    - Given input, output variables can be predicted accurately.

  - **Unsupervised learning** approaches find patterns/relationships/structure in data, but are not optimized to solve a particular predictive task
    - Finds the patterns/relationships/structure in data.
    - NOT optimized to predict

- **Movie recommendation Model:**
  - **Solution 1:** Design a system based on prior knowledge:
    - Cons: depend on *assumptions*, cannot adapt to *new* data
    - Pros: requires no data.
    - Ex: if () case 1; else case 2.
    - USL
  - **Solution 2**: Identify similarity between wall posts and synopses:
    - Cons: depend on *assumptions* about how users relate to items, not be adaptable.
    - Pros: Require non-labeled data
    - USL
  - **Solution 3:** Identify attributes that are associated with positive ratings
    - Cons: requires a (large) dataset of labeled movies
    - Pros: Optimizes a measure we care; easy to adapt to new data
    - SL

- **Regression** (Simple supervised learning approach)
  - **Linear Regression**:
    - (matrix of features as data) * theta = (vector of outputs as label)
    - $X * \theta = y \rightarrow \theta = (X^T X)^{-1} X^T y$
    - Example:
      - Real-valued features: preferences toward certain beers vary with age. (rating = $\theta_0 + \theta_1 *$ age)

==========

Lecture 2

==========

- **Linear Regression Continued:**
  - Model is always linear in Parameter θ, regardless arbitrary combinations of features.
  - Ex1 ~ real valued features:

$$\text{Rating} = \theta_0 + \theta_1 \times \text{ABV} + \theta_2 \times \text{ABV}^2 + \theta_3 \exp(\text{ABV}) + \theta_4 \sin(\text{ABV})$$

  - Ex2 ~ categorical features:
    - **One - Hot encoding**

feature = [1, 0, 0] for "female"
feature = [0, 1, 0] for "other"
feature = [0, 0, 1] for "not specified"

      - To capture n possible categories, we only need (n - 1) dimensions.
  - Ex3 ~ modelling temporal data:
    - $\Theta_0 + \Theta_1 \sin(\text{alpha} + \text{month} * 30)$ is NOT a linear model
    - Apply **piecewise functions**:

$$\text{rating} = \theta_0 + \theta_1 \times \delta(\text{is Feb}) + \theta_2 \times \delta(\text{is Mar}) + \theta_3 \times \delta(\text{is Apr}) \cdots$$

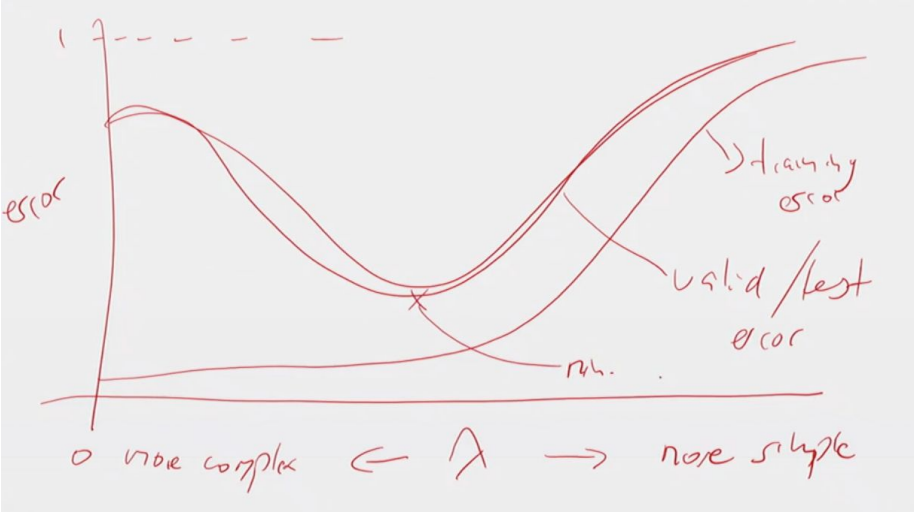
1 if it's Feb, 0 otherwise

- **Regression Diagnostics**
  - **MSE: measure precision**
    - Mean:
      - $\frac{1}{N} \sum_{i=1}^{N} (yi)$
    - Variance:
      - $\frac{1}{N} \sum_{i=1}^{N} (yi - \overline{Y})^2$
    - MSE
      - $\frac{1}{N} \sum_{i=1}^{N} (yi - Xi * \theta)^2$
    - Proportional to the **variance** of the data
    - Reason to use MSE:

        - Small error, small penalty; Huge error, larger penalty.
        - Assumption: distribution of error (Gaosiam? distribution)
    - Coefficient of determination ⇒ R^2
        - Note: if always predict average, MSE = VARi
        - R^2 statistic
            - Fraction of variance Unexplained (FVU)
                - **Measures how fit the regression line is**
            - R^2 = 1 - FVU(f) = 1 - MSE(f) / var(Y)
            - Trivial predictor
                - If $R^2 \rightarrow 0$, it means the feature is **NOT** linearly significant to the dependent variable;
            - Perfect predictor
                - otherwise, $R^2 \rightarrow 1$ the feature **is** linearly significant to the dependent variable
    - MSE can always = 0 (R^2 = 1) by throwing enough random features
    - **MSE & R^2 are used to evaluate how precise the model is.**
    - **A good model is one that generalize to new data.**

- **Overfitting**
    - Definition
        - If a model performs well on training data but does not predict precisely, then this model is overfitting.
    - Occam's Razor
        - θ is the hypothesis
        - The complexity of θ increases as the dimensions grow
            - Simple
                - Type 1: Has few non-zero parameters
                - $||\theta_1||$ = **Σ** θ is small;
                - Type 2: theta is almost uniform
                - $||\theta_2||$ = **Σ** θ^2 is small;
            - Complex
                - Few features are significantly more relevant than others
        - Favor simple model over the complex

- **Regularization**
    - A process of trade-off between the **complexity** and **MSE** (accuracy)
    - $$\arg\min_\theta = \frac{1}{N}\|y - X\theta\|_2^2 + \lambda\|\theta\|_2^2$$
    - Theorems:
        - Lamba increases, training error increases (simpler model)
        - Validation/test error at least the training error (infinitely large)

- Validation/test error usually have “sweet spot”/
- 

- Optimizing the model: Gradient descent
  - Initialize θ at random
  - While Not Converge: $\theta := \theta - \alpha f'(\theta)$

- Model selection
  - lowest training error: MSE is zero, choose high polynomial
  - lowest test error: Cheating, can’t use test set multiple times.
  - **Three → validation set (lambda)**
    - Tune any model parameters that are not directly optimized.

- Sets
  - Training Set: **optimize** the model’s **parameters**
  - Test Set: how well it performs on **unseen data**
  - Validation Set: **tune** any model parameters that are not directly optimized. To choose amongst models.

- **Week 1 Core Ideas:**
  - Regression can be cast in terms of **maximizing a likelihood**.
  - Gradient descent for model optimization
  - Regularization: trade off complexity and accuracy on test
  - Regularization pipeline.

==========

Lecture 3

==========

- **Classification**
  The prediction about binary or categorical variables.

    - Linear classifiers:

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

- **Naive Bayes**:

    - Classify according to which probability is greater than **0.5**
        - **p(label|data) ~ p(-label|data)**

    - Simplifying **Assumption**: conditionally independent.
        - $P(feat_1, feat_2 \mid label) = P(feat_1|label) * P(feat_2|label)$
        - Can be done with counting

    - Posterior = prior * likelihood / evidence.
      P(label | features) = P(label) P(features | label) / P(features)

    - Ex: amazon's children's book. Assumption is false.

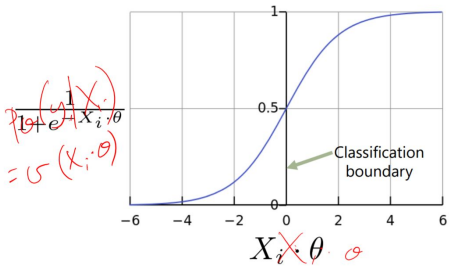
    - Pro/Cons:
        - Simple to compute; Efficient to train
        - Double counting if assumption is false.

- **Logistic Regression:**

    - Traditional linear/probabilistic model: **double counting**, because all parameters sum together.
        - Ex: Height = weight * theta1 + shoe_size * theta2.

    - Convert a real-valued expression into a probability:
        - Sigmoid functions: $f(t) = 1 / (1 + e^{-t})$

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



-
- Map X_i * theta into 0 - 1.
- X_i * theta → large if Y positive; → small if Y negative.

- Optimize :

$$L_\theta(y|X) = \prod_{y_i=1} p_\theta(y_i|X_i) \prod_{y_i=0} (1 - p_\theta(y_i|X_i))$$

  - Take logarithm to convert multiplication into addition.
  - **Subtract** regularizer
  - Compute Gradient
  - Solve using gradient **ascent**

- Generalize:
  - Compute binary classifier for each class.
  - If inconsistent, choose highest confidence

- Disadvantage:
  - Don't optimize **misclassification error**
  - Every instance influences the model.
  - More expensive to train.

- **Support Vector Machine**
  - A classifier that minimizes misclassification (more expensive to train)
  - Idea:
    - The margin to be as wide as possible
    - Penalizing the points on the wrong side
  - **Soft-margin formulation**:

$$\arg\min_{\theta,\alpha,\ \xi_i>0} \frac{1}{2}\|\theta\|_2^2 + C\sum_i \xi_i$$

such that

$$\forall_i y_i(\theta \cdot X_i - \alpha) \geq 1 - \xi_i$$

- **Summary**
  - Above classifiers make decisions by associating **weights** with **features**.
  - **Naive bayes**:
    - Probabilistic model (fits p(label|data))
    - Simple to compute just by counting
    - ++ Easiest to implement, most efficient to "train"
    - ++ If we have a process that generates feature that are independent given the label, it's a very sensible idea
    - -- Otherwise it suffers from a "double-counting" issue
  - **Logistic Regression**
    - Fixes the "double counting" problem present in naïve Bayes
    - 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!)
    - -- More expensive to train
  - **SVMs**

    Non-probabilistic: optimizes the **classification error** rather than the likelihood
  - minimizing the misclassification error
  - -- more expensive to train

==========

Lecture 4

==========

- **Evaluate Classifiers**
  - **Imbalanced** data: Assign additional weight to negative instance if far fewer positive examples than negative ones.
    - TP, TN, F_1 score.
  - Mistake are more **costly in one direction**: False positives are nuisances but false negative are disastrous. E.g. 宁可错误检查一千个平民包，不可放过一个炸弹
    - F_{beta} score.
    - Trade-off between precision and recall
  - **Most confident** predictions: recommender; only care the top ranking results.
    - Precision@k.

- **Definitions:**
  - General Formula: (True/False | Predicted as positive or negative)

		Label	
		true	false
Prediction	true	true positive	false positive
	false	false negative	true negative

  - Classification Accuracy
    - (TP + TN) / All
  - Error Rate
    - (FP + FN) / ALL
  - True Positive Rate =
    - true positives / # labeled positive
    - TP / (TP + FN)
  - True Negative Rate
    - true negative / # labeled negative
    - TN / (TN + FP)
  - Balanced Error Rate
    - ½ (FPR + FNR) = 1 - ½ (TPR + TNR)

- **Optimize a balanced error measure**:

$$L_\theta(y|X) = \prod_{y_i=1} p_\theta(y_i|X_i) \prod_{y_i=0} (1 - p_\theta(y_i|X_i))$$

$$\ell_\theta(y|X) = \sum_{y_i=1} \log \sigma(X_i \cdot \theta) + \sum_{y_i=0} \log(1 - \sigma(X_i \cdot \theta))$$

penalizes for: $y_i=1$ → FN; $y_i=0$ → FP

adjust: $\frac{1}{|y_i=1|} \sum_{y_i=1} \log \sigma(X_i \cdot \theta) + \frac{1}{|y_i=0|} \sum_{y_i=0} \log(1 - \sigma(X_i \cdot \theta))$

-

- **Ranking**
  - Classifiers associate **score** with prediction
  - The predicted labels **don't matter**
  - Positively labeled points tend to be at **higher ranks** than negative ones
  - **Sort** predicted labels by their confidence

- **Precision**
  - In the documents retrieved, how many of them are relevant

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

"fraction of retrieved documents that are relevant"

  -
  - Precision@k: budget of k retrieved documents

- **Recall**
  - In all the relevant documents, how many of them are retrieved.

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

"fraction of relevant documents that were retrieved"

  -
  - Recall@k: budget of k retrieved documents

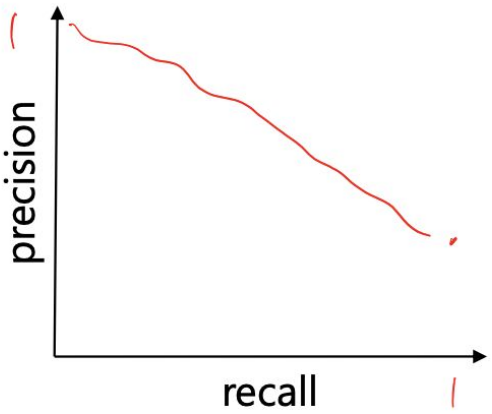
- **Precision & Recall Relationship**
  - Harmonic mean of Precision and recall:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

- Beta: low → precision more important; high → recall more important

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \text{precision} + \text{recall}}$$

- Increase with budget:



==========

Lecture 5

==========

- **Dimension Reduction**:
  - Low dimensional representation of high dimensional data.
  - Data lies (approximately on some low-dimensional space:
  - Data can be compressed.

- Ratings for movies:
  - A1: Sparse vector including all movies
    - Very high dimensional (overfitting)
    - Missing Values
    - Can't add new movies efficiently (re-train the model every new addition)
    - Redundancy
  - A2: Describe preferences using **low-dimensional vector**

- Represent the complete text of a document
  - A1: Sparse vector counting all words
    - Incredibly high-dimensional data
    - Cost of storage and manipulation
    - Redundant encoding
    - Many dimensions devote to "long tail"
  - A2: A low-dimensional vector describing the **topics** in the document

- Connections in social network
  - A1: adjacency matrix
    - Large
    - Fine-grained: can't encode which nodes are familiar
  - A2: represent each node/user in **communities**
    - Only describe the relationships with vector[# of communities]

- **Unsupervised Learning:**
  - Not predictive task, but these techniques are important for future task.
  - Describe the data set.

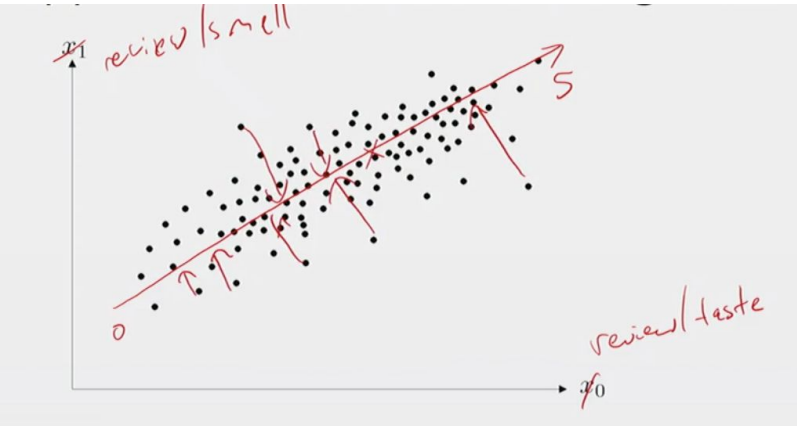
- **Principal Component Analysis**

  - **Select** a few important features
  - **Compress** the data by ignoring meaningless components

  - General Idea:
    - M: number of features; N: number of observations.
    - Signal: $X \sim R^{M \times N} \rightarrow$ Compressed signal $Y' \sim R^{K \times N}$, where $M > K$. And the process to recover from its compressed version: $f(Y') \sim= X$

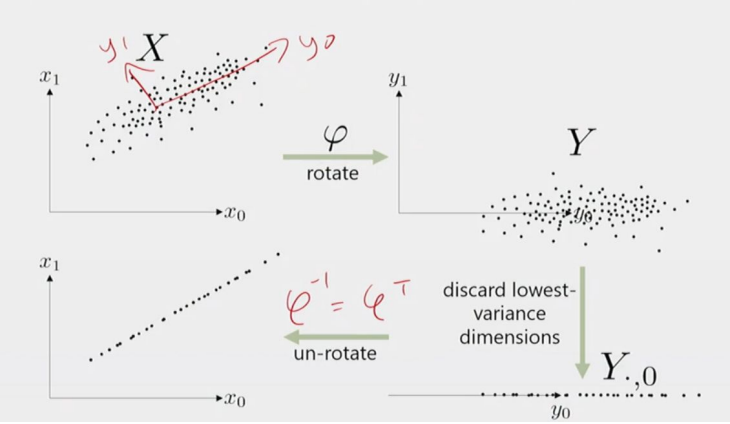
- Keep the dimensions with **HIGHEST variance**, and discard the dimensions with the **lowest variance**; Maximize the amount of "**randomness**" that gets preserved when compressing data.

- Example:
  - Map 2-D data into 1-D.



- Find Important Dimensions**: new basis** for the data
  - Most variance is along x_0
  - Most leftover variance (not explained by x_0) is along x_1
  - Most leftover variance (not explained by x_0, x_1) is along x_2
  - Etc.

  - **Input**: $X \sim R^{M \times N}$; Basis: $R^{M \times M}$. Such that X is rotated.



  - **Objective**: find φ such that **least information** is lost during the process.

  - Math:
    - Complete reconstruction:
      $X = \varphi_1 y_1 + \varphi_2 y_2 + \ldots + \varphi_M y_M = \sum \varphi_j y_j$ from j = 1 to M

    - Approximate reconstruction:
      Delete some data and only keep the first k dimensions:
      $X \sim \sum \varphi_i y_i$ (i from 1 to k) + $\sum \varphi_j b_j$ (j from k + 1 to n, b as a constant).

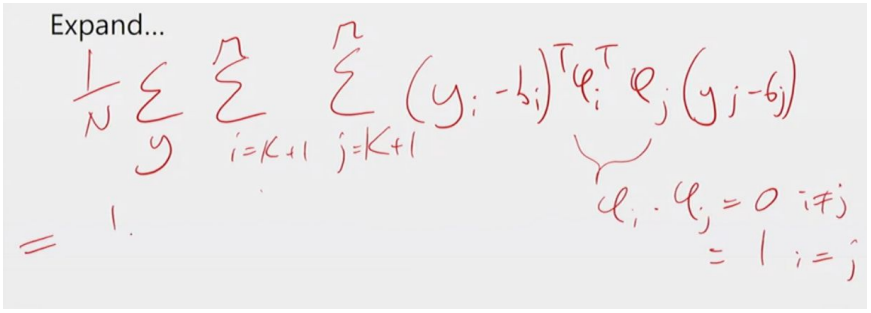
- Minimize the MSE:

$$\min_{\varphi,b} \frac{1}{N} \sum_{y} \left\| \sum_{j=1}^{K} \varphi_j y_j + \sum_{j=K+1}^{M} \varphi_j b_j - \varphi^T y \right\|_2^2$$

- Simplifying:

$$\min_{\varphi,b} \frac{1}{N} \sum_{y} \left\| \sum_{j=K+1}^{M} \varphi_j (y_j - b_j) \right\|_2^2$$

- Expand by the rule: $||x||_2^2 = x^T x$.



- Delete some dimensions, rotate data, and then rotate back, how much information did I lost:

$$\min_{\varphi,b} \frac{1}{N} \sum_{y} \sum_{j=K+1}^{M} (y_j - b_j)^2$$

- Minimize MSE: constant b_j = mean value of y_j; then the equation equals to the **variance** in the discarded dimensions.

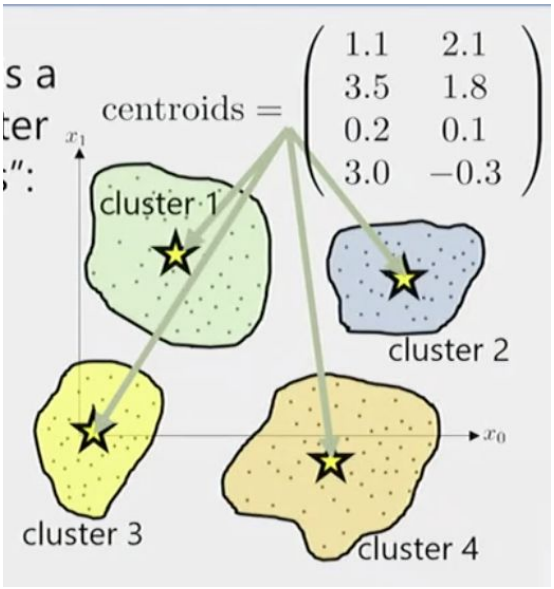
- If we want to optimally (in terms of MSE) project some data into a low dimensional space, we choose the projection by taking the **eigenvectors** corresponding to the **largest eigenvalues of the covariance matrix**.

- Codes:
  - pca = PCA(n_components=5); pca.fit(X);
  - Pca.components:
    - eigenvector, rotation matrix;
    - New basis for the data.
    - PSI.

        - E.g.: Each **row** represents the **single question** you should ask to preserve the most information.
    - Example: Denoise images.

- Summary:
    - **Low-dimensional representation** that best compress or "**summarizes**" our data
    - Keep dimensions with the **highest variance**, and discard dimensions with lower variance. Capture the aspects of the data that are hardest to predict.

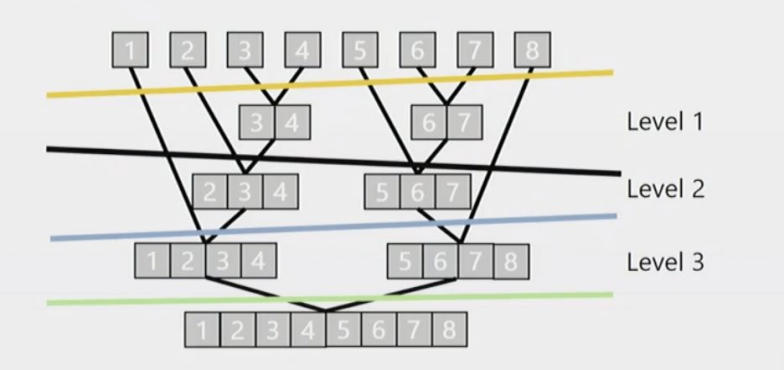
- **Clustering - K means**
    - Idea:
        - For highly clustered data, PCA doesn't work.
        - Input: matrix of features
        - Choose K **centroids**(C) and **cluster assignment** (Y)
        - Output: list of cluster "**centroids**" that minimizes reconstruction error.



    - Optimize:
        - $\text{Min}_{c,\,y}\ \Sigma_i \,||\, X_i - C_{yi} \,||_2^2$, NP Hard Problem.
        - Approximation: Greedy algorithm.
            - Random Initialization
            - Assign each X_i to its nearest centroid
            - Update each centroid to be the mean of assigned points.
            - Repeat
        - **Converge:** each iteration reduces the reconstruction error. And solution is finite.

- **Clustering - Hierarchical**
    - A representation that encodes that points have **some features in common**, but not the others.
    - Algorithm:
        - Hierarchical clustering: gradually fusing clusters whose points are closest together.

-

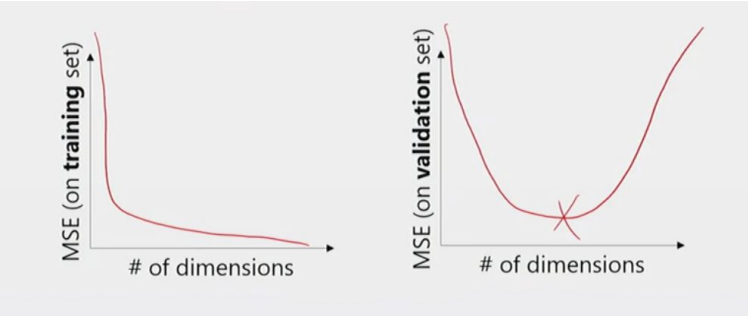


- **Model Selection:** how to choose K in K-means.

  - Method 1: “compressing” our data
    - Choose however many dimensions we can afford to obtain a given file size/compression ratio.
    - Keep adding dimensions until adding more no longer decreases the reconstruction error.

  - Method 2: generating potentially useful features for some other predictive task.
    - Choose however many dimensions results in best **prediction performance on held out data.**

-

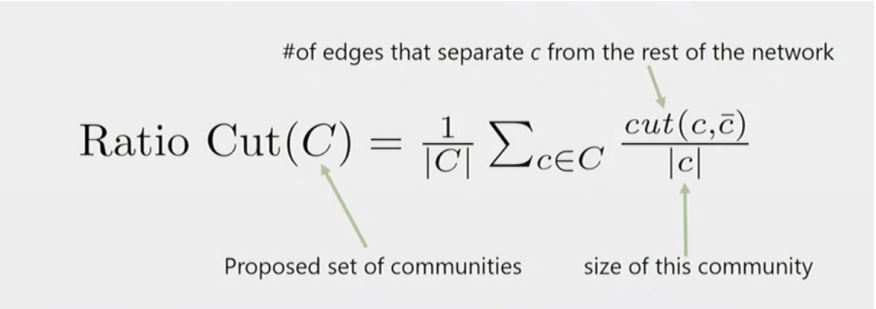


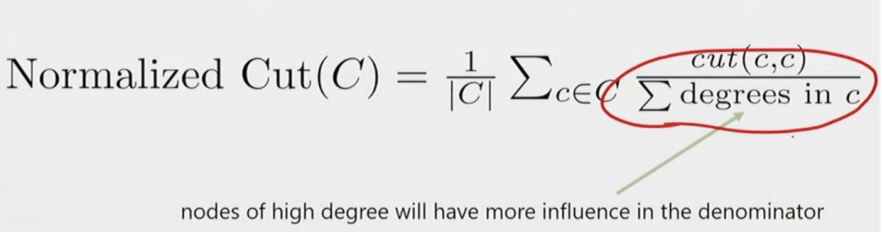
- **Community Detection - Graphs**

  - Clustering: group sets of points based on **features**.
  - Community Detection: group sets of points based on **connectivity.** [adjacency matrix].

  - Community:
    - Connected Components: connected members
    - Minimum Cut: few edges between communities
    - Clique Percolation: cliquishness [mutual connect]
    - Network Modularity: dense inside, few edges outside

  - Method 1: **Connected Components**:
    - Sets of nodes which are reachable from each other
    - Not useful for real graphs: a giant component containing almost all nodes

- Method 2: **Graph Cut:**
  - **Cut** the network into two partitions such that # of edges crossed by the cut is **minimal**.
    - Just cut the graph with empty set.
  - **Ratio cut**: a cut that favors large communities over small one.
    - 

    - Encourage to have equal size.
    - But still end up in the same previous solution.
  - **Normalized Cut**: rather than counting all nodes equally, we give more weight to "influential", **high degree** nodes.
    - 

    - E.g.: Karate Club Graph
    - E.g. computer visions: partition image into different parts.

- Social Communities:
  - Disjoint communities
  - Overlapping communities: groups with some intersection
  - Nested communities: one group within each other

- Method 3: **Clique percolation**:
  - Clique: a fully connected graph.
  - Algorithm based on high degrees of mutual connectivity. [Cliquishness]
  - Measure density among graphs.

  - Code:
    - Given a clique size k
    - Initialize every k-clique as its own community
    - While (two communities have a (k - 1) clique in common:
      - Merge I and J into a single community

- Method 4: **Network Modularity:**
  - Define probabilistic model and evaluate the likelihood of observing a certain set of communities compared to some null model.

- Null model: edges are equally likely between any pair of nodes, regardless of community structure. [random model].

- **Deviate**: how much does a propose community deviate from null.

- Modularity:
  - E_{kk} = fraction of edges in community k
    = # edges within community k / # edges
  - A_k = # edge endpoints in community k / # edge endpoints
    = Fraction that we would expect if randomly allocated.
  - $$e_{kk} = \frac{\#\text{ edges with both endpoints in community } k}{\#\text{ edges}}$$
    $$a_k = \frac{\#\text{ edge endpoints in community } k}{\#\text{ edge endpoints}}$$
    $$Q = \sum_{k=1}^{K} (e_{kk} - a_k^2)$$
  - $$-\frac{1}{2} \leq Q < 1$$
    Far fewer edges in communities than we would expect at random
    Far more edges in communities than we would expect at random

- Algorithm: greedy algorithm.

==========

Lecture 6

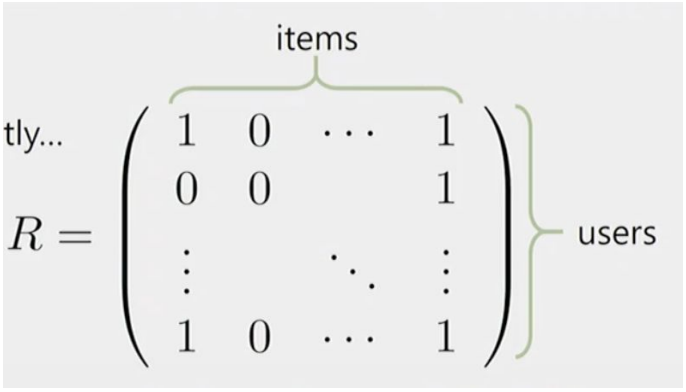
==========

- **Recommender System**: model people's preferences, opinions, behavior
  - Discover new content
  - Find the content we were already looking for
  - Discover which things go together
  - Personalize user experiences in response to user feedback
  - Identify things we like

- Issues with linear predictors:
  - Everyone recommend the same movie
  - Don't model interaction between user and movie
  - F = <user predictor> + <movie predictor>, but they are not independent.

- Recommender: model the **relationships** between people and items.
  - Collaborative filtering: similarity
  - Latent factor model: low dimensional space

- **Collaborative filtering:**
  - Estimate similarity between two users → Items they purchased
  - Estimate similarity between two items → users who purchased them
  - **Unsupervised learning:**
    - Find item i user likes (high rating)
    - Then recommend items similar to i.

  - Definition: represent users and items in matrix
  - 

  - I_u = items purchased by user u; U_i = users who purchased item i.
  - $I_u = \{ i \mid R_{ui} = 1\}$; $U_i = \{ u \mid R_{ui} = 1\}$

  - Method 0: **Euclidean distance**
    - Between two items:

  - $|U_i \setminus U_j| + |U_j \setminus U_i| = || R_i - R_j ||$

- Method 1: **Jaccard Similarity**
  - Normalize the euclidean distance in some sense.
  - Jaccard(U_i , U_j) = $|U_i \cap U_j| / |U_i \cup U_j|$
  - Range:
    - 1 = two users purchased exactly the same items;
    - 0 = two users purchased completely disjoint sets of items.

- Method 2: **Cosine Similarity**
  - cosΘ = A * B / ||A|| ||B|| → $|U_i \cap U_j| / |U_i| * |U_j|$
  - Work for arbitrary **vectors,** can have negative values.
  - Range:
    - 1 (theta = 0): rated by the same users, they all agree
    - -1 (theta = 180): rated by the same users. Completely disagree
    - 0: rated by different sets of users.
  - Cosine similarity (between users):

$$\mathrm{Sim}(u,v) = \frac{\sum_{i \in I_u \cap I_v} R_{u,i} R_{v,i}}{\sqrt{\sum_{i \in I_u \cap I_v} R_{u,i}^2 \sum_{i \in I_u \cap I_v} R_{v,i}^2}} = \frac{A \cdot B}{\|A\| \|B\|}$$

- Method 3: **Pearson correlation**
  - Numerical ratings (rather than thumbs up/thumbs down)
  - Don't want one-star to be the same as five-star
  - Idea: normalize first two real-valued vectors by subtracting their average.
  - items rated by both users; average rating by user *v*

$$\mathrm{Sim}(u,v) = \frac{\sum_{i \in I_u \cap I_v} (R_{u,i} - \bar{R_u})(R_{v,i} - \bar{R_v})}{\sqrt{\sum_{i \in I_u \cap I_v} (R_{u,i} - \bar{R_u})^2 \sum_{i \in I_u \cap I_v} (R_{v,i} - \bar{R_v})^2}}$$

- In practice:
  - Normalize ratings:
    - r(u, i) = 1/z sum (sim(i, j) r_{uj}; More similar, more weight.
    - z = sum sim (i, j)
  - Pro/Cons:
    - Built useful tools out of nothing but **rating data**.
    - One user purchase one item **changes the rankings of every other item** that was purchased by at least one user in common.
    - No use for new users and new items.
    - Won't encourage diverse results.

- **Latent Factor Models**
  - Simplest Possible Model:
    - f(u, i) = α, α = mean of the data → MSE(f) = VAR(R)
  - Second Simplest Model:
    - $f(u, i) = \alpha + \beta_u + \beta_i$,
    - β_u how much does this user tend to rate things above mean
    - B_i how much does this item receive higher ratings than other.
    - Linear model:
      - (u, i) as one hot encoding
      - Theta = <alpha, $beta_u$, $beta_i$ >
      - Doesn't personalize but return only global values.
    - Optimization:
      - $$\arg\min_{\alpha,\beta} \underbrace{\sum_{u,i}(\alpha + \beta_u + \beta_i - R_{u,i})^2}_{\text{error}} + \underbrace{\lambda\left[\sum_u \beta_u^2 + \sum_i \beta_i^2\right]}_{\text{regularizer}}$$
      - **Jointly Convex**
      - Error: minimize; Regularizer: trade-off between complexity and simplicity.
      - Iterative procedure:

$$\alpha = \frac{\sum_{u,i\in\text{train}}(R_{u,i}-(\beta_u+\beta_i))}{N_{\text{train}}}$$

$$\beta_u = \frac{\sum_{i\in I_u} R_{u,i}-(\alpha+\beta_i)}{\lambda+|I_u|}$$

$$\beta_i = \frac{\sum_{u\in U_i} R_{u,i}-(\alpha+\beta_u)}{\lambda+|U_i|}$$

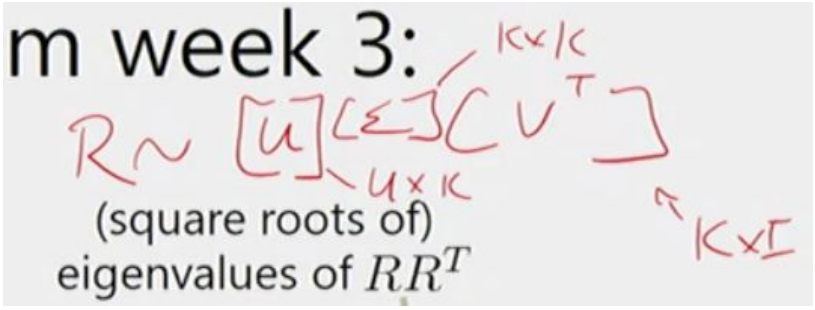
    - Issues
      - Still a function that treats user and items independently
      - Whether a movie good/bad; whether the user/movie generally rates above/below average.

==========

Lecture 7

==========

- **Latent Factor continued**
  - **Dimensionality Reduction**
    - The best rank-k approximation (in terms of MSE): take eigenvectors with the highest eigenvalues.



- Issue: missing ratings. The matrix is undefined.

- Approximate DR with **Gradient Descent**
  - Idea: offset (global average rating) + user/item bias + user/item vector

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

    - R ~ k-d representation of each user * k-d representation of each item
    - $R_{[U \times I]} \sim U_{[\text{each row as user}]} * V^T_{[\text{each column as items}]}$
  - Optimization:

$$\arg\min_{\alpha,\beta,\gamma} \underbrace{\sum_{u,i}(\alpha+\beta_u+\beta_i+\gamma_u\cdot\gamma_i-R_{u,i})^2}_{\text{error}} + \underbrace{\lambda\left[\sum_u \beta_u^2 + \sum_i \beta_i^2 + \sum_i \|\gamma_i\|_2^2 + \sum_u \|\gamma_u\|_2^2\right]}_{\text{regularizer}}$$

    - Regularizer: push the multi-dimensional vector to zero.
  - Issue:
    - **not convex**.
    - Local optimal.
  - Still approximate the equation.
- Observation:
  - Model that uses features only → Model that completely ignores them

- **One Class Recommendation**
  - Dimensionality reduction for **binary** prediction

Past Exams

Spring 15 CSE 190
http://cseweb.ucsd.edu/classes/fa18/cse158-a/files/midterm_cse190_sp15.pdf
Solution:
http://cseweb.ucsd.edu/classes/fa15/cse190-a/slides/lecture10_annotated.pdf

Fall 15 CSE 190
http://cseweb.ucsd.edu/classes/fa18/cse158-a/files/midterm_cse190_fa15.pdf
Solution:
http://cseweb.ucsd.edu/classes/wi17/cse158-a/slides/lecture10_annotated.pdf

Win 17 CSE 158
http://cseweb.ucsd.edu/classes/fa17/cse158-a/files/midterm_cse158_wi17.pdf
Solution:
http://cseweb.ucsd.edu/classes/fa17/cse158-a/slides/midterm_review_annotated.pdf