CSE 150A/250A - Homework 4 Discussion Session

Probabilistic Reasoning and Learning

Today's Agenda

- **Problem 4.1:** Maximum Likelihood Estimation in Belief Networks
- **Problem 4.2:** Markov Modeling
- **Problem 4.3:** Statistical Language Modeling (Coding)

Problem 4.1: Maximum Likelihood Estimation in Belief Networks

Problem Overview

- Two DAGs: G_1 and G_2 with same nodes but **reversed edges**
- Both are chain structures over $X_1, X_2, \ldots, X_n$
- Fully observed dataset with T examples
- Given: $COUNT_i(x)$ and $COUNT_i(x, x')$

Parts (a) & (b): Approach

Key Insight: Maximum Likelihood Estimation for CPTs

- **Identify parent-child relationships in each DAG**
- **For G_1: edges go $X_i \rightarrow X_{i+1}$**
- **For G_2: edges go $X_{i+1} \rightarrow X_i$**
- **Use MLE formula: $P(X|Pa(X)) = \frac{count(X, Pa(X))}{count(Pa(X))}$**

Part (c): Same Joint Distribution?

Goal: Show $P_1^{ML}(X_1, \ldots, X_n) = P_2^{ML}(X_1, \ldots, X_n)$

- Write out the full factorization for each DAG
- Substitute the MLE formulas from parts (a) and (b)
- Simplify using algebra
- Hint: Many terms will cancel out!

Part (d): Graph G_3 - Conditional Independence

Question: Does G_3 also yield the same joint distribution?

- G_3 has **some edges reversed**
- Key concept: Conditional Independence
- Look at node X_{n-2} - how many parents?

Problem 4.2: Markov Modeling

The Setup

Training sequence: S = "aabbbbccddaaddcc"

- **Alphabet: $A = \{a, b, c, d\}$**
- **Length: $L = 16$ tokens**
- **Build two models: Unigram and Bigram**

Part (a): Unigram Model

Likelihood: $P_U(S) = \prod_{\ell=1}^{L} P_1(\tau_\ell)$

- **Count each token's frequency**
- **Divide by total length to get probabilities**
- **Fill in the table!**

Part (b): Bigram Model

Likelihood: $P_B(S) = P_1(\tau_1) \prod_{\ell=2}^{L} P_2(\tau_\ell | \tau_{\ell-1})$

- **Count transitions from token τ to τ'**
- **For each τ, compute** $P_2(\tau' | \tau) = \frac{count(\tau, \tau')}{count(\tau)}$
- **Watch out: some entries will be zero!**

Part (c): Comparing Likelihoods

Strategy: Use qualitative reasoning, not calculations!

- **Unigram model: only cares about token counts**
- **Bigram model: cares about transitions**
- **Unseen bigrams → $P_B = 0$ (log-likelihood = $-\infty$)**
- **Compare sequences with same token counts**

Part (d): Mixture Model

Model: $P_M(\tau'|\tau) = (1-\lambda)P_1(\tau') + \lambda P_2(\tau'|\tau)$

- $\lambda = 0$ → **pure unigram**
- $\lambda = 1$ → **pure bigram**
- **Match each sequence to the correct qualitative behaviour**
- **Consider: does bigram help or hurt?**

Problem 4.3: Statistical Language Modeling (Coding)

Overview: Real-World Language Modeling

Goal: Build models of English text

- **Vocabulary: 500 frequently occurring tokens**
- **Special token: $\langle UNK \rangle$ for unknown words**
- **Data files: unigram counts, bigram counts, vocabulary**
- **Same concepts as Problem 4.2, just bigger!**

Functions 2 & 3: Computing Probabilities

Unigram: $P_u(w) = \frac{count(w)}{\sum_{w'} count(w')}$ **Bigram:** $P_b(w'|w) = \frac{count(w,w')}{\sum_{w''} count(w,w'')}$

- **For each word w, normalize its successors**
- **Same as Problem 4.2!**

Parts (a) & (b): What to Print

Part (a): Print words starting with "M"

- Filter your unigram dictionary
- Return word and probability (as per return format)

Part (b): Top 10 words after "THE"

- Look up bigram_probs["THE"]
- Sort by probability (descending)
- Return top 10(as per return format)

Function 4: compute_sentence_log_likelihood()

Key steps:

- **Split sentence into words**
- **Replace out-of-vocabulary words with $\langle UNK \rangle$**
- **Bigram: use $\langle s \rangle$ as start token, check if bigram exists**
- **Critical: What would happen if bigram missing ?**
- **Make sure to use SUM of log probabilities**

Use the property

$$\log(\boldsymbol{A} * \boldsymbol{B} * \boldsymbol{C} * \boldsymbol{D}) = \log(\boldsymbol{A}) + \log(\boldsymbol{B}) + \log(\boldsymbol{C}) + \log(\boldsymbol{D})$$

Parts (c) & (d): Two Test Sentences

Part (c): "The stock market fell by one hundred points last week"

- All bigrams are in training data
- Compare unigram vs bigram log-likelihood

Part (d): "The sixteen officials sold fire insurance"

- Some bigrams are missing from training
- What happens to bigram log-likelihood?

Function 5: compute_mixed_likelihood()

Formula: $P_m(w'|w) = \lambda P_u(w') + (1 - \lambda) P_b(w'|w)$

- **If bigram doesn't exist, use 0 for** $P_b(w'|w)$
- **This fixes the problem from part (d)!**
- **Compute mixed probability for each word**
- **Return sum of log probabilities**

Function 6: plot_and_get_optimal_lambda()

Part (e): Find the best λ

- **Try $\lambda = \mathbf{0.00, 0.01, 0.02, \ldots, 1.00}$**
- **Compute mixed likelihood for each value**
- **Find the λ that maximizes likelihood**
- **Optional: plot the results (helpful for understanding!)**

Important Coding Tips

- Don't modify function names or signatures!
- Use sum of logs, not log of products (avoid underflow)
- Use relative paths, not absolute paths
- The autograder tests each function independently
- Test locally with the provided data files first

Key Takeaways

- **MLE for Belief Networks:** Use parent-child structure to write CPTs
- **Markov Models:** Balance between simplicity (unigram) and context (bigram)
- **Language Modeling:** Real-world application of probability theory
- **Mixture Models:** Combine strengths of different approaches

Questions?