Introduction to Latent Sequences & Expectation Maximization

CMSC 473/673
UMBC
• EMBEDDINGS/DISTRIBUTED REPRESENTATIONS
• COURSE SO FAR

REVIEW
Neural Language Models

**given some context...**

create/use “distributed representations”...

**combine these representations...**

compute beliefs about what is likely...

**predict the next word**

\[ p(w_i | w_{i-3}, w_{i-2}, w_{i-1}) \propto \text{softmax}(\theta_{w_i} \cdot f(w_{i-3}, w_{i-2}, w_{i-1})) \]
(Some) Properties of Embeddings

Capture “like” (similar) words

<table>
<thead>
<tr>
<th>target</th>
<th>Redmond</th>
<th>Havel</th>
<th>ninjutsu</th>
<th>graffiti</th>
<th>capitulate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Redmond Wash.</td>
<td></td>
<td>Vaclav Havel</td>
<td>ninja</td>
<td>spray paint</td>
<td>capitulation</td>
</tr>
<tr>
<td>Redmond Washington</td>
<td></td>
<td>president Vaclav Havel</td>
<td>martial arts</td>
<td>graffiti</td>
<td>capitted</td>
</tr>
<tr>
<td>Microsoft</td>
<td></td>
<td>Velvet Revolution</td>
<td>swordsmanship</td>
<td>taggers</td>
<td>capitated</td>
</tr>
</tbody>
</table>

Capture relationships

\[
\text{vector('king') - vector('man')} + \text{vector('woman')} \approx \text{vector('queen')}
\]

\[
\text{vector('Paris') - vector('France')} + \text{vector('Italy')} \approx \text{vector('Rome')}
\]
Four kinds of vector models

Sparse vector representations

1. Mutual-information weighted word co-occurrence matrices

Dense vector representations:

2. Singular value decomposition/Latent Semantic Analysis
3. Neural-network-inspired models (skip-grams, CBOW)
4. Brown clusters

Learn more in:
- Your project
- Paper (673)
- Other classes (478/678)
Shared Intuition

Model the meaning of a word by “embedding” in a vector space

The meaning of a word is a vector of numbers

Contrast: word meaning is represented in many computational linguistic applications by a vocabulary index (“word number 545”) or the string itself
Intrinsic Evaluation: Cosine Similarity

Divide the dot product by the length of the two vectors
\[ \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| |\vec{b}|} \]
This is the cosine of the angle between them
\[ \vec{a} \cdot \vec{b} = |\vec{a}| |\vec{b}| \cos \theta \]
\[ \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| |\vec{b}|} = \cos \theta \]

Are the vectors parallel?
-1: vectors point in opposite directions
+1: vectors point in same directions
0: vectors are orthogonal
Course Recap So Far

Basics of Probability

Requirements to be a distribution
(“proportional to”, \( \propto \))

Definitions of conditional probability, joint probability, and independence

Bayes rule, (probability) chain rule
Course Recap So Far

Basics of Probability

- Requirements to be a distribution ("proportional to", $\propto$)
- Definitions of conditional probability, joint probability, and independence
- Bayes rule, (probability) chain rule

Basics of language modeling

Goal: model (be able to predict) and give a score to language (whole sequences of characters or words)

- Simple count-based model
- Smoothing (and why we need it): Laplace (add-$\lambda$), interpolation, backoff
- Evaluation: perplexity
Course Recap So Far

Basics of Probability
- Requirements to be a distribution ("proportional to", \(\propto\))
- Definitions of conditional probability, joint probability, and independence
- Bayes rule, (probability) chain rule

Basics of language modeling
- Goal: model (be able to predict) and give a score to language (whole sequences of characters or words)
- Simple count-based model
- Smoothing (and why we need it): Laplace (add-\(\lambda\)), interpolation, backoff
- Evaluation: perplexity

Tasks and Classification (use Bayes rule!)
- Posterior decoding, noisy channel model
- Evaluations: accuracy, precision, recall, and \(F_\beta\) (\(F_1\)) scores
- Naïve Bayes (given the label, generate/explain each feature independently) and connection to language modeling
Course Recap So Far

Basics of Probability
- Requirements to be a distribution ("proportional to", $\propto$)
- Definitions of conditional probability, joint probability, and independence
- Bayes rule, (probability) chain rule

Basics of language modeling
- Goal: model (be able to predict) and give a score to language (whole sequences of characters or words)
  - Simple count-based model
  - Smoothing (and why we need it): Laplace (add-$\lambda$), interpolation, backoff
  - Evaluation: perplexity

Tasks and Classification (use Bayes rule!)
- Posterior decoding, noisy channel model
- Evaluations: accuracy, precision, recall, and $F_\beta$ ($F_1$) scores
  - Naïve Bayes (given the label, generate/explain each feature independently) and connection to language modeling

Maximum Entropy Models
- Meanings of feature functions and weights
- Use for language modeling or conditional classification ("posterior in one go")
- How to learn the weights: gradient descent
Course Recap So Far

Basics of Probability
- Requirements to be a distribution (“proportional to”, ∝)
- Definitions of conditional probability, joint probability, and independence
- Bayes rule, (probability) chain rule

Basics of language modeling
- Goal: model (be able to predict) and give a score to language (whole sequences of characters or words)
  - Simple count-based model
  - Smoothing (and why we need it): Laplace (add-λ), interpolation, backoff
- Evaluation: perplexity

Tasks and Classification (use Bayes rule!)
- Posterior decoding, noisy channel model
- Evaluations: accuracy, precision, recall, and \( F_\beta \) (\( F_1 \)) scores
- Naïve Bayes (given the label, generate/explain each feature independently) and connection to language modeling

Maximum Entropy Models
- Meanings of feature functions and weights
- Use for language modeling or conditional classification (“posterior in one go”)
- How to learn the weights: gradient descent

Distributed Representations & Neural Language Models

What embeddings are and what their motivation is

A common way to evaluate: cosine similarity
Course Recap So Far

Basics of Probability
- Requirements to be a distribution (“proportional to”, ∝)
- Definitions of conditional probability, joint probability, and independence
- Bayes rule, (probability) chain rule

Basics of language modeling
- Goal: model (be able to predict) and give a score to language (whole sequences of characters or words)
- Simple count-based model
- Smoothing (and why we need it): Laplace (add-λ), interpolation, backoff
- Evaluation: perplexity

Tasks and Classification (use Bayes rule!)
- Posterior decoding, noisy channel model
- Evaluations: accuracy, precision, recall, and $F_\beta$ ($F_1$) scores
- Naïve Bayes (given the label, generate/explain each feature independently) and connection to language modeling

Maximum Entropy Models
- Meanings of feature functions and weights
- Use for language modeling or conditional classification ("posterior in one go")
- How to learn the weights: gradient descent

Distributed Representations & Neural Language Models
- What embeddings are and what their motivation is
- A common way to evaluate: cosine similarity
LATENT SEQUENCES AND LATENT VARIABLE MODELS
Is Language Modeling “Latent?”

\[
p(\text{Colorless green ideas sleep furiously}) = \\
p(\text{Colorless}) * \\
p(\text{green} | \text{Colorless}) * \\
p(\text{ideas} | \text{Colorless green}) * \\
p(\text{sleep} | \text{green ideas}) * \\
p(\text{furiously} | \text{ideas sleep})
\]
Is Language Modeling “Latent?”
Most* of What We’ve Discussed: Not Really

\[
p(\text{Colorless green ideas sleep furiously}) = p(\text{Colorless}) \times p(\text{green | Colorless}) \times p(\text{ideas | Colorless green}) \times p(\text{sleep | green ideas}) \times p(\text{furiously | ideas sleep})
\]

* Neural language modeling as an exception
Three people have been fatally shot, and five people, including a mayor, were seriously wounded as a result of a Shining Path attack today against a community in Junín department, central Peruvian mountain region.
Three people have been fatally shot, and five people, including a mayor, were seriously wounded as a result of a Shining Path attack today against a community in Junín department, central Peruvian mountain region.

Is Document Classification “Latent?”
As We’ve Discussed

\[
\arg\max_X \prod_i p(Y_i|X) * p(X)
\]

\[
\arg\max_X \frac{\exp(\theta \cdot f(x,y))}{Z(x)} * p(x)
\]

\[
\arg\max_X \exp(\theta \cdot f(x,y))
\]
Three people have been fatally shot, and five people, including a mayor, were seriously wounded as a result of a Shining Path attack today against a community in Junin department, central Peruvian mountain region.

These values are unknown but the generation process (explanation) is transparent.
Ambiguity → Part of Speech Tagging

British Left Waffles on Falkland Islands

*British Left Waffles on Falkland Islands*

Adjective  Noun  Verb

*British Left Waffles on Falkland Islands*

Noun  Verb  Noun
observed text
orthography
morphology
lexemes
syntax
semantics
pragmatics
discourse

Adapted from Jason Eisner, Noah Smith
Latent Modeling

explain what you see/annotate

with things “of importance” you don’t

observed text
orthography
morphology
lexemes
syntax
semantics
pragmatics
discourse

Adapted from Jason Eisner, Noah Smith
Latent Sequence Models: Part of Speech

\[ p(\text{British Left Waffles on Falkland Islands}) \]
Latent Sequence Models: Part of Speech

(i): Adjective Noun Verb Prep Noun Noun

(ii): Noun Verb Noun Prep Noun Noun

$p(\text{British Left Waffles on Falkland Islands})$
Latent Sequence Models: Part of Speech

\[ p(\text{British Left Waffles on Falkland Islands}) \]

1. Explain this sentence as a sequence of (likely?) latent (unseen) tags (labels)

(i): Adjective  Noun  Verb  Prep  Noun  Noun
(ii): Noun  Verb  Noun  Prep  Noun  Noun
Latent Sequence Models: Part of Speech

1. Explain this sentence as a sequence of (likely?) latent (unseen) tags (labels)

2. Produce a tag sequence for this sentence

p(British Left Waffles on Falkland Islands)

(i): Adjective  Noun  Verb  Prep  Noun  Noun
(ii): Noun  Verb  Noun  Prep  Noun  Noun
$p(X \mid Y) \propto p(Y \mid X) \ast p(X)$

possible (clean) output

observed (noisy) text

translation/decode model

(clean) language model

Decode

Rerank
Latent Sequence Model: Machine Translation

\[ p(X \mid Y) \propto p(Y \mid X) \ast p(X) \]

possible (clean) output

observed (noisy) text

(clean) language model

translation/decode model

Decode

Rerank
Latent Sequence Model: Machine Translation

Le chat est sur la chaise.
Latent Sequence Model: Machine Translation

Le chat est sur la chaise.

The cat is on the chair.
Latent Sequence Model: Machine Translation

How do you know what words translate as?
Learn the translations!

Le chat est sur la chaise.

The cat is on the chair.
Latent Sequence Model: Machine Translation

How do you know what words translate as?
Learn the translations!

How?
Learn a “reverse” latent alignment model
\[ p(\text{French words, alignments} \mid \text{English words}) \]

Le chat est sur la chaise.
The cat is on the chair.
Latent Sequence Model: Machine Translation

How do you know what words translate as?
Learn the translations!

How?
Learn a “reverse” latent alignment model
\( p(\text{French words, alignments} | \text{English words}) \)

Alignment?
Words can have different meaning/senses

Le chat est sur la chaise.

The cat is on the chair.
Latent Sequence Model: Machine Translation

*How do you know what words translate as?*

Learn the translations!

*How?*

Learn a “reverse” latent alignment model

\[ p(\text{French words, alignments} \mid \text{English words}) \]

*Alignment?*

Words can have different meaning/senses

*Why Reverse?*

\[ p(\text{English} \mid \text{French}) \propto p(\text{French} \mid \text{English}) \times p(\text{English}) \]
How to Learn With Latent Variables (Sequences)

Expectation Maximization
Example: Unigram Language Modeling

\[ p(w_1, w_2, \ldots, w_N) = p(w_1)p(w_2) \cdots p(w_N) = \prod_{i} p(w_i) \]
Example: Unigram Language Modeling

\[ p(w_1, w_2, \ldots, w_N) = p(w_1)p(w_2) \cdots p(w_N) = \prod_{i} p(w_i) \]

maximize (log-)likelihood to learn the probability parameters
Example: Unigram Language Modeling with Hidden Class

\[ p(w_1, w_2, ..., w_N) = p(w_1)p(w_2) \cdots p(w_N) = \prod_i p(w_i) \]

\[ p(z_1, w_1, z_2, w_2, ..., z_N, w_N) = p(z_1)p(w_1|z_1) \cdots p(z_N)p(w_N|z_N) \]
\[ = \prod_i p(w_i|z_i)p(z_i) \]
Example: Unigram Language Modeling with Hidden Class

\[
p(w_1, w_2, \ldots, w_N) = p(w_1)p(w_2) \cdots p(w_N) = \prod_i p(w_i)
\]

\[
p(z_1, w_1, z_2, w_2, \ldots, z_N, w_N) = p(z_1)p(w_1|z_1) \cdots p(z_N)p(w_N|z_N)
= \prod_i p(w_i|z_i)p(z_i)
\]

examples of latent classes \(z\):
- part of speech tag
- topic ("sports" vs. "politics")
Example: Unigram Language Modeling with Hidden Class

\[ p(w_1, w_2, \ldots, w_N) = p(w_1)p(w_2) \cdots p(w_N) = \prod_i p(w_i) \]

add complexity to better explain what we see

\[ p(z_1, w_1, z_2, w_2, \ldots, z_N, w_N) = p(z_1)p(w_1|z_1) \cdots p(z_N)p(w_N|z_N) \]

= \prod_i p(w_i|z_i) p(z_i)

goal: maximize (log-)likelihood
Example: Unigram Language Modeling with Hidden Class

\[ p(w_1, w_2, ..., w_N) = p(w_1)p(w_2) \cdots p(w_N) = \prod_i p(w_i) \]

\[ p(z_1, w_1, z_2, w_2, ..., z_N, w_N) = p(z_1)p(w_1|z_1) \cdots p(z_N)p(w_N|z_N) \]

\[ = \prod_i p(w_i|z_i) p(z_i) \]

goal: maximize (log-)likelihood

we don’t actually observe these \( z \) values

we just see the words \( w \)
Example: Unigram Language Modeling with Hidden Class

\[ p(w_1, w_2, ..., w_N) = p(w_1)p(w_2) \cdots p(w_N) = \prod_i p(w_i) \]

we don't actually observe these \( z \) values
we just see the words \( w \)

\[ p(z_1, w_1, z_2, w_2, ..., z_N, w_N) = p(z_1)p(w_1|z_1) \cdots p(z_N)p(w_N|z_N) \]

\[ = \prod_i p(w_i|z_i) p(z_i) \]

goal: maximize (log-)likelihood

we don't actually observe these \( z \) values
we just see the words \( w \)

if we \textit{did} observe \( z \), estimating the probability parameters would be easy...

but we don’t! :(
Example: Unigram Language Modeling with Hidden Class

\[ p(w_1, w_2, ..., w_N) = p(w_1)p(w_2) \cdots p(w_N) = \prod_i p(w_i) \]

add complexity to better explain what we see

\[ p(z_1, w_1, z_2, w_2, ..., z_N, w_N) = p(z_1)p(w_1|z_1) \cdots p(z_N)p(w_N|z_N) \]

\[ = \prod_i p(w_i|z_i)p(z_i) \]

goal: maximize (log-)likelihood

we don’t actually observe these \( z \) values
we just see the words \( w \)

if we did observe \( z \), estimating the probability parameters would be easy...
but we don’t! :(  

if we knew the probability parameters then we could estimate \( z \) and evaluate likelihood... but we don’t! :(
Example: Unigram Language Modeling with Hidden Class

\[ p(z_1, w_1, z_2, w_2, ..., z_N, w_N) = p(z_1)p(w_1|z_1) \cdots p(z_N)p(w_N|z_N) \]

\[ = \prod_i p(w_i|z_i) p(z_i) \]

we don’t actually observe these \( z \) values

goal: maximize \textit{marginalized} (log-)likelihood
Example: Unigram Language Modeling with Hidden Class

\[ p(z_1, w_1, z_2, w_2, \ldots, z_N, w_N) = p(z_1)p(w_1|z_1) \cdots p(z_N)p(w_N|z_N) \]

\[ = \prod_{i} p(w_i|z_i) \, p(z_i) \]

we don’t actually observe these \( z \) values

goal: maximize marginalized (log-)likelihood
Example: Unigram Language Modeling with Hidden Class

\[
p(z_1, w_1, z_2, w_2, \ldots, z_N, w_N) = p(z_1)p(w_1|z_1) \cdots p(z_N)p(w_N|z_N) \\
= \prod_{i} p(w_i|z_i) p(z_i)
\]

*we don’t actually observe these z values*

goal: maximize **marginalized** (log-)likelihood
Marginal(ized) Probability

\[ p(w) = p(z_1, w) + p(z_2, w) + p(z_3, w) + p(z_4, w) \]
Marginal(ized) Probability

\[ p(w) = p(z_1, w) + p(z_2, w) + p(z_3, w) + p(z_4, w) = \sum_{z=1}^{4} p(z_i, w) \]
Marginal(ized) Probability

\[ p(w) = \sum_{z} p(z, w) \]
Marginal(ized) Probability

\[ p(w) = \sum_{z} p(z, w) \]

\[ = \sum_{z} p(z)p(w \mid z) \]
Example: Unigram Language Modeling with Hidden Class

\[ p(z_1, w_1, z_2, w_2, \ldots, z_N, w_N) = p(z_1)p(w_1|z_1) \cdots p(z_N)p(w_N|z_N) \]

\[ = \prod_i p(w_i|z_i) \, p(z_i) \]

we don’t actually observe these \( z \) values

goal: maximize *marginalized* (log-)likelihood

\[
p(w_1, w_2, \ldots, w_N) = \left( \sum_{z_1} p(z_1, w) \right) \left( \sum_{z_2} p(z_2, w) \right) \cdots \left( \sum_{z_N} p(z_N, w) \right)
\]
Example: Unigram Language Modeling with Hidden Class

\[ p(z_1, w_1, z_2, w_2, ..., z_N, w_N) = p(z_1) p(w_1 | z_1) \cdots p(z_N) p(w_N | z_N) \]

goal: maximize \textit{marginalized} (log-)likelihood

\[ p(w_1, w_2, ..., w_N) = \left( \sum_{z_1} p(z_1, w) \right) \left( \sum_{z_2} p(z_2, w) \right) \cdots \left( \sum_{z_N} p(z_N, w) \right) \]

if \textit{we did} observe \( z \), estimating the probability parameters would be easy...

but we don’t! :(  

if \textit{we knew} the probability parameters then we could estimate \( z \) and evaluate likelihood... but we don’t! :(
if we knew the probability parameters
then we could estimate $z$ and evaluate likelihood... but we don’t! :(

if we did observe $z$, estimating the
probability parameters would be easy...
but we don’t! :( 

if we did not know the probability parameters estimating the probability parameters would be easy... but we don’t! :(

if we knew the probability parameters then we could estimate $z$ and evaluate likelihood... but we don’t! :(
Expectation Maximization (EM)

0. Assume *some* value for your parameters

Two step, iterative algorithm

1. E-step: count under uncertainty (compute expectations)

2. M-step: maximize log-likelihood, assuming these uncertain counts
Expectation Maximization (EM): E-step

0. Assume *some* value for your parameters

Two step, iterative algorithm

1. E-step: count under uncertainty, assuming these parameters

\[
p(z_i)
\]

\[
\rightarrow
\]

\[
\text{count}(z_i, w_i)
\]

2. M-step: maximize log-likelihood, assuming these uncertain counts
**Expectation Maximization (EM): E-step**

0. Assume *some* value for your parameters

Two step, iterative algorithm

1. **E-step**: count under uncertainty, assuming these parameters

   \[ p(z_i) \rightarrow \text{count}(z_i, w_i) \]

   We’ve already seen this type of counting, when computing the gradient in maxent models.
Expectation Maximization (EM): M-step

0. Assume *some* value for your parameters

Two step, iterative algorithm

1. E-step: count under uncertainty, assuming these parameters

2. M-step: maximize log-likelihood, assuming these uncertain counts
\[
\max_{\theta} \mathbb{E}_{z \sim p_\theta(t) \cdot |w)} \left[ \log p_\theta(z, w) \right]
\]
EM Math

\[
\max_{\theta} \mathbb{E}_{z \sim p_{\theta}(t) (\cdot | w)} \left[ \log p_{\theta}(z, w) \right]
\]

*E-step: count under uncertainty*

*M-step: maximize log-likelihood*
EM Math

$\max_{\theta} \mathbb{E}_z \sim p_{\theta(t)}(\cdot|w) \left[ \log p_{\theta}(z, w) \right]$

*E-step: count under uncertainty*

*old parameters*  
*posterior distribution*

*M-step: maximize log-likelihood*
EM Math

\[
\max_{\theta} \mathbb{E}_{z \sim p_{\theta(t)}(\cdot | w)} \left[ \log p_{\theta}(z, w) \right]
\]

E-step: count under uncertainty

\[
\max_{\theta} \mathbb{E}_{z \sim p_{\theta(t)}(\cdot | w)} \left[ \log p_{\theta}(z, w) \right]
\]

M-step: maximize log-likelihood
Imagine three coins

Flip 1\textsuperscript{st} coin (penny)

If heads: flip 2\textsuperscript{nd} coin (dollar coin)

If tails: flip 3\textsuperscript{rd} coin (dime)
Imagine three coins

Flip 1\textsuperscript{st} coin (\textit{penny})

If heads: flip 2\textsuperscript{nd} coin (\textit{dollar coin})

If tails: flip 3\textsuperscript{rd} coin (\textit{dime})

only observe these (record heads vs. tails outcome)

don’t observe this
Imagine three coins

If heads: flip 2\textsuperscript{nd} coin (dollar coin)

If tails: flip 3\textsuperscript{rd} coin (dime)

unobserved: vowel or consonant? part of speech?

observed: a, b, e, etc.
We run the code, vs. The run failed
Three Coins/Unigram With Class Example

Imagine three coins

Flip 1\textsuperscript{st} coin (penny)

If heads: flip 2\textsuperscript{nd} coin (dollar coin)

If tails: flip 3\textsuperscript{rd} coin (dime)

\[ p(\text{heads}) = \lambda \quad \text{vs.} \quad p(\text{tails}) = 1 - \lambda \]

\[ p(\text{heads}) = \gamma \quad \text{vs.} \quad p(\text{tails}) = 1 - \gamma \]

\[ p(\text{heads}) = \psi \quad \text{vs.} \quad p(\text{tails}) = 1 - \psi \]
Three Coins/Unigram With Class Example

Imagine three coins

\[ p(\text{heads}) = \lambda \]
\[ p(\text{tails}) = 1 - \lambda \]

\[ p(\text{heads}) = \gamma \]
\[ p(\text{tails}) = 1 - \gamma \]

\[ p(\text{heads}) = \psi \]
\[ p(\text{tails}) = 1 - \psi \]

Three parameters to estimate: \( \lambda \), \( \gamma \), and \( \psi \)
If all flips were observed

\[
\begin{align*}
p(\text{heads}) &= \lambda & p(\text{heads}) &= \gamma & p(\text{heads}) &= \psi \\
p(\text{tails}) &= 1 - \lambda & p(\text{tails}) &= 1 - \gamma & p(\text{tails}) &= 1 - \psi
\end{align*}
\]
Three Coins/Unigram With Class Example

If all flips were observed

\[ p(\text{heads}) = \lambda \quad p(\text{heads}) = \gamma \quad p(\text{heads}) = \psi \]
\[ p(\text{tails}) = 1 - \lambda \quad p(\text{tails}) = 1 - \gamma \quad p(\text{tails}) = 1 - \psi \]

\[ p(\text{heads}) = \frac{4}{6} \quad p(\text{heads}) = \frac{1}{4} \quad p(\text{heads}) = \frac{1}{2} \]
\[ p(\text{tails}) = \frac{2}{6} \quad p(\text{tails}) = \frac{3}{4} \quad p(\text{tails}) = \frac{1}{2} \]
Three Coins/Unigram With Class Example

But not all flips are observed $\rightarrow$ set parameter values

\[ p(\text{heads}) = \lambda = .6 \quad p(\text{heads}) = .8 \quad p(\text{heads}) = .6 \]
\[ p(\text{tails}) = .4 \quad p(\text{tails}) = .2 \quad p(\text{tails}) = .4 \]
Three Coins/Unigram With Class Example

But not all flips are observed → set parameter values

\[
p(\text{heads}) = \lambda = 0.6 \quad p(\text{heads}) = 0.8 \quad p(\text{heads}) = 0.6
\]
\[
p(\text{tails}) = 0.4 \quad p(\text{tails}) = 0.2 \quad p(\text{tails}) = 0.4
\]

Use these values to compute posteriors

\[
p(\text{heads} \mid \text{observed item } H) = \frac{p(\text{heads} \& H)}{p(H)}
\]
\[
p(\text{heads} \mid \text{observed item } T) = \frac{p(\text{heads} \& T)}{p(T)}
\]
Three Coins/Unigram With Class Example

But not all flips are observed → set parameter values

\[ p(\text{heads}) = \lambda = 0.6 \quad p(\text{heads}) = 0.8 \quad p(\text{heads}) = 0.6 \]
\[ p(\text{tails}) = 0.4 \quad p(\text{tails}) = 0.2 \quad p(\text{tails}) = 0.4 \]

Use these values to compute posteriors

\[ p(\text{heads} \mid \text{observed item H}) = \frac{p(H \mid \text{heads})p(\text{heads})}{p(H)} \]

rewrite joint using Bayes rule

marginal likelihood
Three Coins/Unigram With Class Example

But not all flips are observed $\rightarrow$ set parameter values

$$p(\text{heads}) = \lambda = 0.6 \quad p(\text{heads}) = 0.8 \quad p(\text{heads}) = 0.6$$
$$p(\text{tails}) = 0.4 \quad p(\text{tails}) = 0.2 \quad p(\text{tails}) = 0.4$$

Use these values to compute posteriors

$$p(\text{heads} \mid \text{observed item H}) = \frac{p(H \mid \text{heads})p(\text{heads})}{p(H)}$$

$$p(H \mid \text{heads}) = 0.8 \quad p(T \mid \text{heads}) = 0.2$$
Three Coins/Unigram With Class Example

### Flips

<table>
<thead>
<tr>
<th>H</th>
<th>H</th>
<th>T</th>
<th>H</th>
<th>T</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>T</td>
<td>H</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

But not all flips are observed → *set* parameter values

\[
p(\text{heads}) = \lambda = .6 \quad p(\text{heads}) = .8 \quad p(\text{heads}) = .6
\]

\[
p(\text{tails}) = .4 \quad p(\text{tails}) = .2 \quad p(\text{tails}) = .4
\]

Use these values to compute posteriors

\[
p(\text{heads} | \text{observed item H}) = \frac{p(H | \text{heads})p(\text{heads})}{p(H)}
\]

\[
p(H | \text{heads}) = .8 \quad p(T | \text{heads}) = .2
\]

\[
p(H) = p(H | \text{heads}) * p(\text{heads}) + p(H | \text{tails}) * p(\text{tails})
\]

\[
= .8 * .6 + .6 * .4
\]
Three Coins/Unigram With Class Example

Use posteriors to update parameters

\[
\begin{align*}
p(\text{heads} | \text{obs. } H) &= \frac{p(H| \text{heads})p(\text{heads})}{p(H)} \\
&= \frac{.8 \times .6}{.8 \times .6 + .6 \times .4} \approx 0.667
\end{align*}
\]

\[
\begin{align*}
p(\text{heads} | \text{obs. } T) &= \frac{p(T| \text{heads})p(\text{heads})}{p(T)} \\
&= \frac{.2 \times .6}{.2 \times .6 + .6 \times .4} \approx 0.334
\end{align*}
\]

(in general, \(p(\text{heads} | \text{obs. } H)\) and \(p(\text{heads} | \text{obs. } T)\) do NOT sum to 1)
Three Coins/Unigram With Class Example

Use posteriors to update parameters

\[
p(\text{heads} \mid \text{obs. H}) = \frac{p(H \mid \text{heads})p(\text{heads})}{p(H)}
= \frac{.8 \times .6}{.8 \times .6 + .6 \times .4} \approx 0.667
\]

\[
p(\text{heads} \mid \text{obs. T}) = \frac{p(T \mid \text{heads})p(\text{heads})}{p(T)}
= \frac{.2 \times .6}{.2 \times .6 + .6 \times .4} \approx 0.334
\]

(in general, \(p(\text{heads} \mid \text{obs. H})\) and \(p(\text{heads} \mid \text{obs. T})\) do NOT sum to 1)

\[p(\text{heads}) = \frac{\# \text{ heads from penny}}{\# \text{ total flips of penny}}\]

fully observed setting

our setting: partially-observed

\[p(\text{heads}) = \frac{\# \text{ expected heads from penny}}{\# \text{ total flips of penny}}\]
Three Coins/Unigram With Class Example

Use posteriors to update parameters

\[
p(\text{heads} | \text{obs. H}) = \frac{p(H|\text{heads})p(\text{heads})}{p(H)} = \frac{.8 \times .6}{.8 \times .6 + .6 \times .4} \approx 0.667
\]

\[
p(\text{heads} | \text{obs. T}) = \frac{p(T|\text{heads})p(\text{heads})}{p(T)} = \frac{.2 \times .6}{.2 \times .6 + .6 \times .4} \approx 0.334
\]

our setting: partially-observed

\[
p^{(t+1)}(\text{heads}) = \frac{\text{# expected heads from penny}}{\text{# total flips of penny}} = \mathbb{E}_{p^{(t)}}[\text{# expected heads from penny}] \frac{\text{# total flips of penny}}{\text{# total flips of penny}}
\]
Three Coins/Unigram With Class Example

Use posteriors to update parameters

\[
p(\text{heads} \mid \text{obs. H}) = \frac{p(H) \cdot p(\text{heads})}{p(H)} = \frac{.8 \cdot .6}{.8 \cdot .6 + .6 \cdot .4} \approx 0.667
\]

\[
p(\text{heads} \mid \text{obs. T}) = \frac{p(T) \cdot p(\text{heads})}{p(T)} = \frac{.2 \cdot .6}{.2 \cdot .6 + .6 \cdot .4} \approx 0.334
\]

\[
p^{(t+1)}(\text{heads}) = \frac{\text{# expected heads from penny}}{\text{# total flips of penny}}
= \frac{\mathbb{E}_{p(t)}[\text{# expected heads from penny}]}{\text{# total flips of penny}}
= \frac{2 \cdot p(\text{heads} \mid \text{obs. H}) + 4 \cdot p(\text{heads} \mid \text{obs. T})}{6} \approx 0.444
\]

our setting: partially-observed
Expectation Maximization (EM)

0. Assume *some* value for your parameters

Two step, iterative algorithm:

1. E-step: count under uncertainty *(compute expectations)*

2. M-step: maximize log-likelihood, assuming these uncertain counts
Related to EM

Latent clustering

K-means:  
https://www.csee.umbc.edu/courses/undergraduate/473/f18/kmeans/

Gaussian mixture modeling