Midterm sample questions

UMass CS 585, Fall 2015

October 18, 2015

1 Midterm policies

The midterm will take place during lecture next Tuesday, 1 hour and 15 minutes.

It is closed book, EXCEPT you can create a 1-page "cheat sheet" for yourself with any notes you like. One page front and back. Feel free to collaborate to create these notes. You will probably find the studying implicit in the act of creating the notes is even more useful than actually having them.

2 Topics on the midterm

Language concepts

- Parts of speech
- The Justeson-Katz noun phrase patterns

Probability / machine learning

- Probability theory: Marginal probs, conditional probs, law(s) of total probability, Bayes Rule.
- Maximum likelihood estimation
- Naive Bayes
- Relative frequency estimation and pseudocount smoothing
- Logistic regression (for binary classification)
- Perceptron
- Averaged Perceptron

Structured models

- Hidden Markov models
- Viterbi algorithm
- Log-linear models and CRFs
- Structured Perceptron

3 Bayes Rule

You are in a noisy bar diligently studying for your midterm, and your friend is trying to get your attention, using only a two word vocabulary. She has said a sentence but you couldn't hear one of the words:

$$(w_1 = \text{hi}, w_2 = \text{yo}, w_3 = ???, w_4 = \text{yo})$$

Question 1. Assume that your friend was generating words from this first-order Markov model:

$$\begin{array}{llll} p(\text{hi}|\text{hi}) & = 0.7 & p(\text{yo}|\text{hi}) & = 0.3 \\ p(\text{hi}|\text{yo}) & = 0.5 & p(\text{yo}|\text{yo}) & = 0.5 \end{array}$$

Given these parameters, what is the posterior probability of whether the missing word is "hi" or "yo"?

[Solution: This question is asking for $p(w_3|w_1, w_2, w_4)$. By the Markov assumption we can ignore w_1 completely, thus just $p(w_3|w_2, w_4)$. Next: we want to manipulate this into a form where we can apply our model parameters, which specify $p(w_t|w_{t-1})$ for any pair of wordtypes (those four numbers above). Our model does not tell us $p(w_3|w_2, w_4)$, nor does it tell us $p(w_3|w_4)$ ... but it *does* tell us $p(w_3|w_2)$ and $p(w_4|w_3)$. We can start to get the second from $p(w_3|w_2, w_4)$ by applying Bayes Rule to flip w_3 and w_4. (This is an instances of background-conditional Bayes Rule: $P(a|bc) = P(b|ac)P(a|c)/P(b|c)$, which is like normal Bayes Rule except there's a "background" variable c always hanging on the right side.)

So we use Bayes Rule where the prior is $p(w_3|w_2 = yo)$ (a function of w_3) and the likelihood is $p(w_4 = yo|w_3)$ (a function of w_3).

$$p(w_3|w_2, w_4) = (1/Z)p(w_3|w_2)p(w_4|w_2, w_3) \quad (1)$$
$$p(w_3|w_2, w_4) = (1/Z)p(w_3|w_2)p(w_4|w_3) \text{ by Markov assumption} \quad (2)$$
$$p(?? = hi) = (1/Z)p(hi|yo)p(yo|hi) = (1/Z)(0.5)(0.3) = (1/Z)0.15 \quad (3)$$
$$p(?? = yo) = (1/Z)p(yo|yo)p(yo|yo) = (1/Z)(0.5)(0.5) = (1/Z)0.25 \quad (4)$$
$$Z = 0.15 + 0.25 = 0.4 \quad (5)$$
$$p(?? = hi) = 15/40 \quad (6)$$
$$p(?? = yo) = 25/40 \quad (7)$$

I find it easiest to think of Z as summing over all possible versions of the denominator: $Z = \sum_{w_3} p(w_3|w_2 = yo)p(w_4 = yo|w_3)$. You could also start with $Z = p(w_3|w_4)$ then use the sum rule to work it out from there.]

Question 2. The following questions concern the basic pseudocount smoothing estimator we used in problem set 1.

1. Pseudocounts should only be added when you have lots of training data. True or False? **[Solution:** F — even with lots of training data, you always have rare words. If "bat" appears once in sports and never in non-sports, do you really want

$p(bat|nonsports) = 0$? If so, any document that contains "bat" can never be classified as sports. That is extreme. Using pseudocounts alleviates this.]

2. Pseudocounts should be added only to rare words. The count of common words should not be changed. True or False? [**Solution:** F — note pseudocounts have a smaller effect on common words.]

3. What happens to Naive Bayes document posteriors (for binary classification), if you keep increasing the pseudocount parameter really really high? [HINT: you can try to do this intuitively. It may help to focus on the $P(w|y)$ terms. A rigorous approach is to use L'Hospital's rule.]

   (a) They all become either 0 or 1.

   (b) They all become 0.5.

   (c) They all become the prior [NOTE Oct18: this option added to solutions]

   (d) Neither of the above.

[**Solution:** They all become the prior. the easy way to see this is, imagine a giant alpha like a million or a zillion. for any word w,

$$p(w|y) = \frac{n_{w,y} + \alpha}{n_y + V\alpha} = \frac{n_{w,y} + 1,000,000}{n_y + V1,000,000} \rightarrow \frac{\alpha}{V\alpha} = \frac{1}{V}$$

where $n_{w,y}$ is the number of tokens among doc class y that are wordtype w, and n_y is the number of tokens for doc class y. those two numbers are dominated by the giant α, which causes all words to have the same uniform probability. OK so consider the posterior ratio (using token notation here),

$$\frac{p(y=1 \mid \vec{w})}{p(y=0 \mid \vec{w})} = \frac{p(y=1)}{p(y=0)} \frac{p(w_1|y=1)}{p(w_1|y=0)} \frac{p(w_2|y=1)}{p(w_2|y=0)} \frac{p(w_3|y=1)}{p(w_3|y=0)} \ldots \tag{8}$$

$$= \frac{p(y=1)}{p(y=0)} \frac{1/V}{1/V} \frac{1/V}{1/V} \frac{1/V}{1/V} \ldots \tag{9}$$

$$= \frac{p(y=1)}{p(y=0)} \tag{10}$$

that implies $p(y=1|\vec{w}) = p(y=1)$. (Exercise: show this)]

4 Classification

We seek to classify documents as being about sports or not. Each document is associated with a pair $(\vec{x}, y)$, where $\vec{x}$ is a feature vector of word counts of the document and y is the label for whether it is about sports ($y = 1$ if yes, $y = 0$ if false). The vocabulary is size 3, so feature vectors look like $(0, 1, 5)$, $(1, 1, 1)$, etc.

4.1 Naive Bayes

Consider a naive Bayes model with the following conditional probability table:

word type	1	2	2
$P(w \mid y = 1)$	1/10	2/10	7/10
$P(w \mid y = 0)$	5/10	2/10	3/10

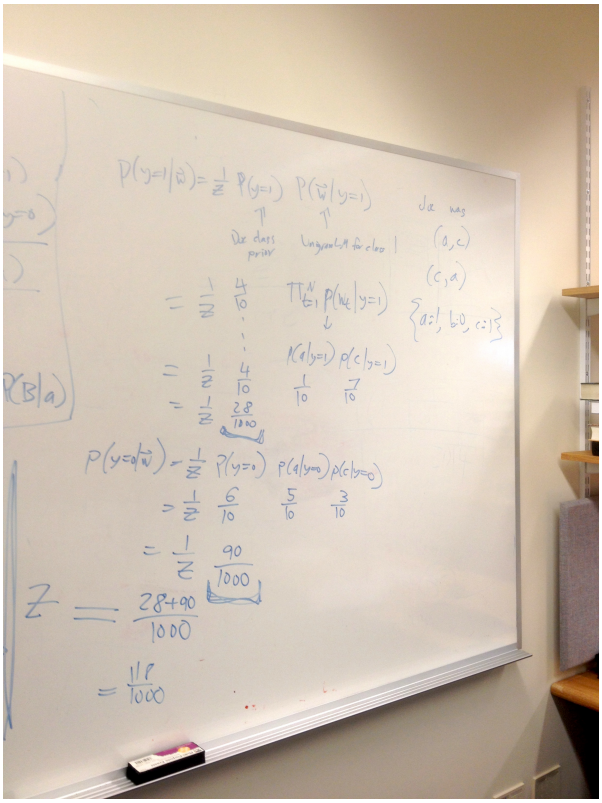
and the following prior probabilities over classes:

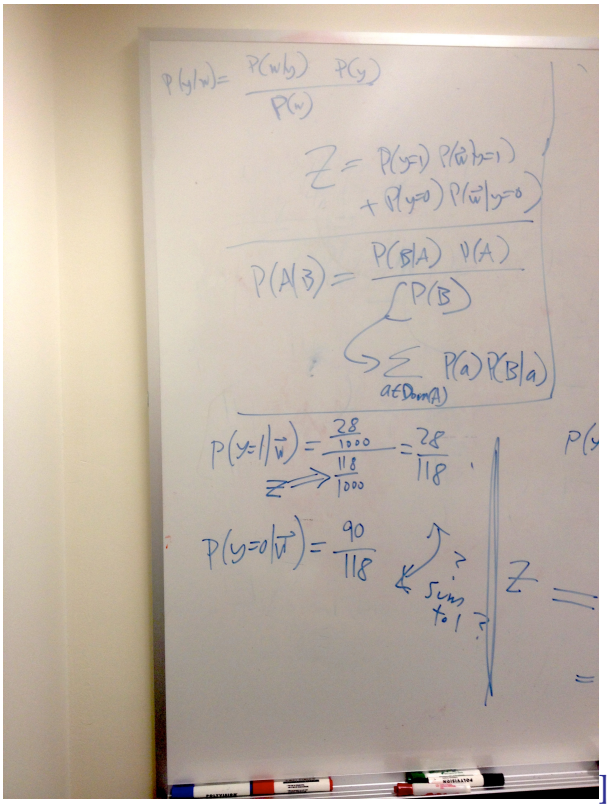
$P(y = 1)$	$P(y = 0)$
4/10	6/10

Question 3.
Consider the document with counts $\vec{x} = (1, 0, 1)$.

1. Which class has highest posterior probability?
2. What is the posterior probability that the document is about sports?

[Solution:





]

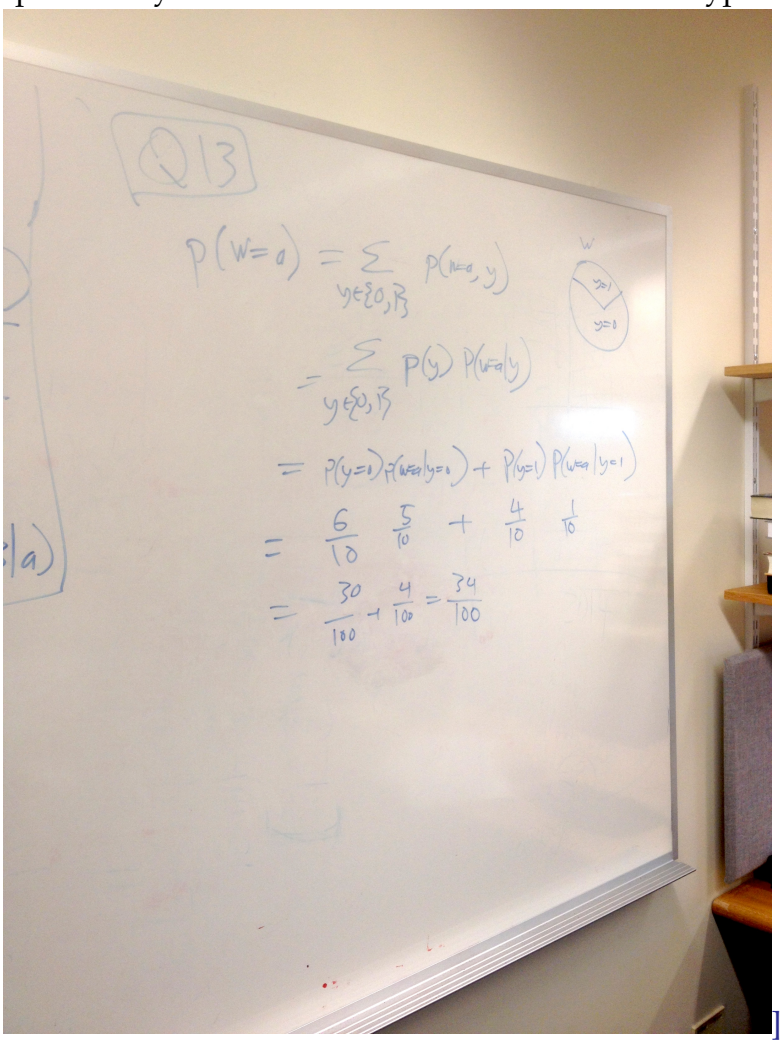
Question 4. Consider the document with counts $\vec{x} = (2, 0, 1)$. Is it the case that $P(y = 1 \mid \vec{x} = (2, 0, 1)) = P(y = 1 \mid \vec{x} = (1, 0, 1))$? If not, please calculate for $(2, 0, 1)$.

Question 5. In lectures, and in the JM reading, we illustrated Naive Bayes in terms of TOKEN generation. However, $\vec{x}$ is WORD COUNTS, i.e. the BOW vector. Please rewrite the unnormalized log posterior $P(y = 1 \mid doc)$ in terms of $\vec{x}$, instead of in terms of each word token as in lecture.

Question 6.

1. Suppose that we know a document is about sports, i.e. $y = 1$. True or False, the Naive Bayes model is able to tell us the probability of seeing $x = (0, 1, 1)$ under the model.
2. If True, what is the probability?

Question 7. Now suppose that we have a new document that we don't know the label of. What is the probability that a word in the document is wordtype 1?

[**Solution:** 

]

Question 8. True or False: if the Naive Bayes assumption holds for a particular dataset (i.e., that the feature values are independent of each other given the class label) then no other model can achieve higher accuracy on that dataset than Naive Bayes. Explain. [Update: this question is too weird. Bad question.] [**Solution:** this question is really subtle. if a generative model is actually true, then you should do inference with the model and it should beat out other ones. so if NB is true, it should beat logreg or perceptron. on the other hand, more than just the NB conditional indep assumption is at play ... for example there's also the pseudocount, which we treat as a fixed hyperparam. typically, discriminative models – like logreg or perceptron – beat generative ones like NB just because NB is set up not in a way that is true or even trying to be true, but in a way that is convenient to compute.]

Question 9. Can Naive Bayes be considered a log linear model? If so, explain why; if not, example why not. [**Solution:** Yes. $\log p(y)$ log-priors are one type of feature (one for each class). And $\log p(w|y)$ log-word-probs are another type of feature (one for each wordtype and class). And use a $\vec{x}$ representation based on BOW word counts. then $\log p(y|x) = C + \theta^\mathsf{T} f(x, y)$ where θ is organized to have both types of logprobs in it in one big vector, and $f(x, y)$ is organized to give an indicator feature for which class y is, and counts for all the y versions of each word. and C is the normalizing constant $C = -\log \sum_{y'} \exp \theta^\mathsf{T} f(x, y')$.]

Question 10. Show that for Naive Bayes with two classes, the decision rule $f(x)$ can be written in terms of $\frac{\log[P(y=1|x)]}{\log[P(y=0|x)]}$. Can the decision rule be formulated similarly for multiclass Naive Bayes? [**Solution:** No. the posterior ratio only can compare two classes. in multiclass NB, you could use a ratio to compare wehther, say, class 2 is more likely than class 5. but to compute teh argmax over all classes, you gotta iterate through all of them.]

Question 11. In terms of exploratory data analysis, why might it be interesting and important to compute the log odds of various features? [**Solution:** ranking words by raw count within a class just shows you words that are common to both, like grammatical function words. these dont tell you much about the nature of the language or content for that class. by contrast, ranking by log-odds gives you words that are unique to a particular class, which are often more interesting. on the other hand, this might overemphasize rare words.]

4.2 Logistic Regression

Question 12. Consider a logistic regression model with weights $\beta = (0.5, 0.25, 1)$. A given document has feature vector $x = (1, 0, 1)$. NOTE: for this problem you will be exponentiating certain quantities. You do not need to write out your answer as a number, but instead in terms of exp() values, e.g., P = 1 + 2exp(−1).

1. What is the probability that the document is about sports?
2. What is the probability that it is not about sports?

[**Solution:** use the equation from ari's lecture]

Question 13. Consider a logistic regression model with weights $\beta = (-ln(4), ln(2), -ln(3))$. A given document has feature vector $x = (1, 1, 1)$. Now, please provide your answer in the form of a fraction $\frac{a}{b}$.

1. What is the probability that the document is about sports?

[**Solution:** this uses logs for the weights just to make the math easier. also note in this class we always use natural logs at least up to now. the exp of a sum is the product of the exp of the terms which simplifies nicely here

$$a = \exp \beta^{\mathsf{T}} x = \exp(-\log(4) + \log(2) - \log(3)) = e^{-\log 4} e^{\log 2} e^{-\log 3} = \frac{1}{4} \times 2 \times \frac{1}{3} = \frac{1}{6}$$

$$p(y = 1|x) = a/[1 + a] = \frac{1/6}{7/6} = \frac{1}{7}$$

]

Question 14. Consider a logistic regression model with weights $\beta = (\beta_1, \beta_2, \beta_3)$. A given document has feature vector $x = (1, 0, 1)$.

1. What is a value of the vector β such that the probability of the document being about sports is 1 (or incredibly close)? [**Solution:** make weights on feature 1 or 3 be like +10,000]

2. What is a value of the vector β such that the probability of the document being about sports is 0 (or incredibly close)? [**Solution:** make weights on feature 1 or 3 be like -10,000]

Question 15. Consider the following two weight vectors for logistic regression:

- $w = (10000, -2384092, 24249, 284924, -898)$
- $w' = (1.213, -.123, 2.23, 3.4, -2)$

For which of these weight vectors is small changes between test instances likely to make large changes in classification? Which of these models do you think generalizes better and why?

5 Language stuff

Question 16. Each of the following sentences has an incorrect part-of-speech tag. Identify which one and correct it. (If you think there are multiple incorrect tags, choose the one that is the most egregious.) We'll use a very simple tag system:

- NOUN – common noun or proper noun
- PRO – pronoun
- ADJ – adjective
- ADV – adverb
- VERB – verb, including auxiliary verbs
- PREP – preposition
- DET – determiner
- X – something else

1. Colorless/ADV green/ADJ clouds/PRO sleep/VERB furiously/ADV ./X **[Solution:** clouds/NOUN]
2. She/PRO saw/VERB herself/PRO through/PREP the/ADJ looking/ADJ glass/NOUN ./X **[Solution:** the/DET]
3. Wait/NOUN could/VERB you/PRO please/X ?/X **[Solution:** Wait/VERB]

6 Perceptron

Question 17. In HW2 we saw an example of when the averaged perceptron outperforms the vanilla perceptron. There is another variant of the perceptron that often outperforms the vanilla perceptron. This variant is called the **voting perceptron**. Here's how the voting perceptron works:

- initialize the weight vector
- if the voting perceptron misclassifies an example at iteration i, update the weight vector and store it as w_i.
- if it makes a correct classification at iteration i, do not update the weight vector but store w_i anyway.
- To classify an example with the voting perceptron, we classify that example with each w_i and tally up the number of votes for each class. The class with the most votes is the prediction.

Despite often achieving high accuracy, the voting perceptron is rarely used in practice. Why not? **[Solution:** The voting perceptron stores every single weight vector computed. This takes $O(T * |W|)$ space to store where T is the number of iterations we train and $|W|$ is the size of the weight vector. This can be huge for many normal problems as opposed to the averaged perceptron which only require $O(|W|)$ space to store its weight vector. Similarly, the averaged perceptron can make predictions in linear time in the size of the weight vector; the voting perceptron only makes predictions in time linear in $T * |W|$ which can be much larger.]

Question 18. [NOTE: we won't ask for any proofs by induction on the test]
Recall that the averaged perceptron algorithm is as follows:

- Initialize $t = 1, \theta_0 = \vec{0}, S_0 = \vec{0}$
- For each example i (iterating multiples times through dataset),
  - Predict $y^* = \arg\max_{y'} \theta^\mathsf{T} f(x_i, y')$
  - Let $g_t = f(x_i, y_i) - f(x_i, y^*)$
  - Update $\theta_t = \theta_{t-1} + r g_t$
  - Update $S_t = S_{t-1} + (t-1) r g_t$
  - $t := t + 1$
- Return $\bar{\theta}_t = \theta_t - \frac{1}{t} S_t$

Use proof by induction to show this algorithm correctly computes the average weight vector for any t, i.e.,

$$\frac{1}{t} \sum_{i=1}^{t} \theta_t = \theta_t - \frac{1}{t} S_t$$

Question 19. For the case of the averaged perceptron, why don't we make predictions during training with the averaged weight vector? [**Solution:** This is a bad question. the answer is, that's just not what the algo is. you're supposed to make predictions with the current raw weight vector and you want it to flop around a lot, then at the end average over all of them. theory and practice says this is good, i guess.]

Question 20. Why wouldn't we want to use the function below to update the weight vector when training a percetron?

```
def update_weights(weight_vec, gradient):
  updated_weights = defaultdict(float)
  for feat, weight in weight_vec.iteritems():
    updated_weights[feat] += weight
  for feat, weight in gradient.iteritems():
    updated_weights[feat] += weight
  return updated_weights
```

[**Solution:** This function creates a new weight vector that contains the sum of the old weight vector and the gradient. Copying over this entire vector is slow. Instead we should update the weights in place.]

7 HMM

Consider an HMM with 2 states, A, B and 2 possible output variables $\Delta, \square$, with transition and emission probabilities from HW2. All probabilities statements are implicitly conditioning on $s_0 = START$.

Question 21. Explain the difference between
$P(s_1 = A \mid o_2 = \Delta)$ versus $P(s_1 = A \mid o_2 = \Delta, s_3 = END)$

Question 22. Rewrite $P(s_1 \mid o_2)$ so that you could calculate it for any particular values of s_1 and o_2. (This is like in HW2, except you should be able to do it abstractly without the numbers or particular values and swap in the numbers only at the end.) **[Solution:** Note that $p(s_1|o_2) \neq p(s_1)$ because knowledge about future affects knowledge about the past, even though it's not how the model generates. the math illustrates this but make sure you get it intuitively. knowing o_2 tells you what s_2 could have been, which affects what s_1 could have been.

You need to apply the sum rule to get out s_2: sum over all paths to get from s_1 to o_2. Also need to apply bayes rule at some point, since $s_1|o_2$ is the wrong direction. in lecture and OH i sometimes did the sum rule first, and it eventually works but gets a little nasty. so instead let's try doing bayes rule first.

$$p(s_1|o_2) = p(o_2|s_1)p(s_1)/p(o_2)$$

ok now hit the lik term and the denom with sum rule. do them separately to minimize mistakes.

$$p(o_2|s_1) = \sum_{s_2} p(o_2|s_2, s_1)p(s_2|s_1) = \sum_{s_2} p(o_2|s_2)p(s_2|s_1)$$

the simplification is applying the HMM conditional indep assumption. once you know the hidden state above o_2, knowing the past doesnt give any additional information. next the denom. sum out all paths.

$$p(o_2) = \sum_{s_2} p(o_2|s_2)p(s_2) \qquad = \sum_{s_2} p(o_2|s_2) \sum_{s_1} p(s_2|s_1)p(s_1) \tag{11}$$

these are now all in terms of HMM model parameters, so you can plug in numbers to evaluate. we won't make you do nasty arithmetic on the midterm test because math is the worst]

Question 23. Rewrite $P(s_1 \mid o_2, s_3 = END)$ so that you could calculate it for any particular values of s_1 and o_2.

Question 24. Why does the END state matter?

Question 25. (Here's what HW2 1.3 was supposed to be.)
Is it the case that $P(o_2 = \Delta \mid s_1 = A) = P(o_2 = \Delta \mid s_1 = A, s_3 = A)$?

Question 26. Write an expression that computes the probability of the HMM emitting the sequence $\Delta, \square$ given that the first state is A and the length of the sequence is 2 (remember to consider the start and end states).

[Solution:

$$P(o_1 = \Delta, o_2 = \square | s_1 = A, s_3 = End, s_0 = Start) = \frac{P(o_1 = \Delta, o_2 = \square, s_1 = A, s_3 = End, s_0 = Start)}{P(s_1 = A, s_3 = End, s_0 = Start)}$$
$$= \frac{\sum_{x \in \{A,B\}} P(o_1 = \Delta | s_1 = A) P(s_2 = x | s_1 = A) P(o_2 = \square | s_2 = x) P(End | s_2 = x)}{\sum_{x' \in \{A,B\}} P(s_2 = x' | s_1 = A) P(End | s_2 = x')}$$

]

8 Viterbi

Question 27. Here's a proposal to modify Viterbi to use less memory: for each token position t, instead of storing all $V_t[1]..V_t[K]$, instead store one probability, for the best path so far. Can we compute an optimal solution in this approach? Why or why not? **[Solution:** This is actually just the greedy algorithm. It won't work because at $t-1$ you can't consider all K possible prefix endings at t. Sometimes, what would be best at t will no longer be best when you consider what happens at $t+1$. The trick of Viterbi is you only need to consider neighboring timesteps, not longer timesteps, to consider all relevant possibilities.]

Question 28. Here's an erroneous version of the (multiplicative-version) Viterbi algorithm. The line in the inner loop had

- BUGGY: $V_t[k] := \max_j V_{t-1}[j] P_{trans}(k \mid j) P_{emit}(w_t \mid j)$
- CORRECT: $V_t[k] := \max_j V_{t-1}[j] P_{trans}(k \mid j) P_{emit}(w_t \mid k)$

Please describe one specific issue that the buggy version of this code would have. For example, describe an important thing in the data that the buggy version ignores.

Question 29. Consider the Eisner ice cream HMM (from J&M 3ed ch 7, Figure 7.3), and a sequence of just one observation, $\vec{w} = (3)$. There are only 2 possible sequences, (HOT) or (COLD). Calculate both their joint probabilities ($p(w, y)$). Which sequence is more likely?
[Solution: (HOT) is optimal]

Question 30. Now consider the observation sequence $\vec{w} = (3, 1, 1)$. Perform the Viterbi algorithm on paper, stepping through it and drawing a diagram similar to Figure 7.10. What is the best latent sequence, and what is its probability? To check your work, try changing the first state; is the joint probability better or worse? (To really check your work you could enumerate all 8 possibilities and check their probabilities, but that is not fun without a computer.)

[**Solution:** Actually (HOT, COLD, COLD) is optimal. This is not a good example.

Better: consider the sequences (1) and (1,1). The first gets $y = (HOT)$ while the second gets $y = (COLD, COLD)$. Viterbi selects the alternative all-COLD path after it considers the second timestep evidence.]

Question 31. Compare how the Viterbi analyzed this sequence, in contrast to what a greedy algorithm would have done. Is it different? Why? Why is this a different situation than the previous example of $\vec{w} = (3)$?

[**Solution:** for the original sequences (3) and (3,1,1) this question doesn't make sense.

for (1) and (1,1) it's interesting: given more data, Viterbi changed its provisional answer to the first timestep. The greedy algorithm would not be able to do this. See the Viterbi notes on the website.]