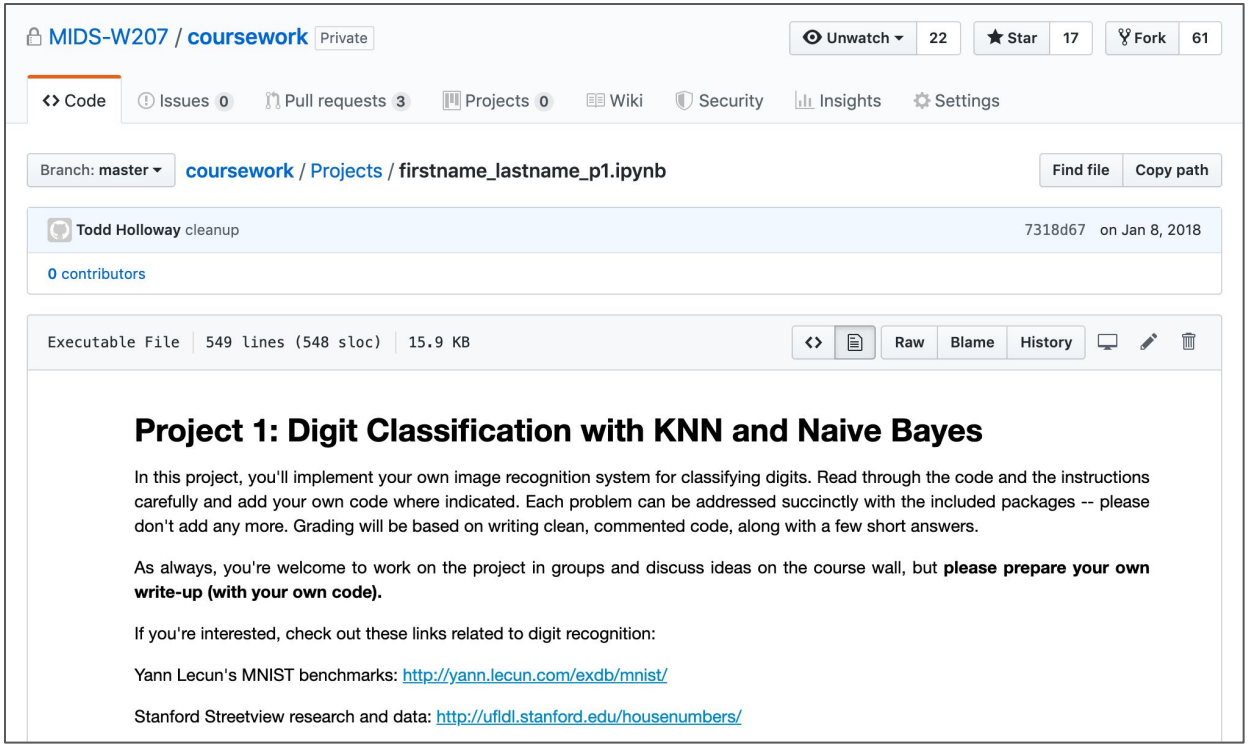
Week 3

Agenda

1. Async Review
2. Graham article discussion
3. Norvig article discussion
4. Finish Domingos paper discussion
5. Time-permitting: Notebooks

For next week: Start project 1

Project 1



Due Sunday, June 9th

Quizzes as Warm-up

What makes naive Bayes "naive"?

- It usually doesn't work very well.
- The assumption that the classes are independent.
- The assumption that the features are independent given the class.

Feature selection is important because

- It can remove poorly estimated features.
- It keeps only the features that give the optimal performance.

Summing log probabilities is equivalent to multiplying probabilities.

- True
- False

A perfectly calibrated classifier is ____% accurate on examples where the posterior probability is 85%.

What is the Laplace (with k=1) smoothed estimate for P(sun) given this data: domain: {sun,rain,wind} observations: [sun,rain,rain,wind,sun,sun]

In our one-feature spam classifier, we have made no assumptions of independence.

- True
- False

Naive Bayes Review

Bayes's Rule

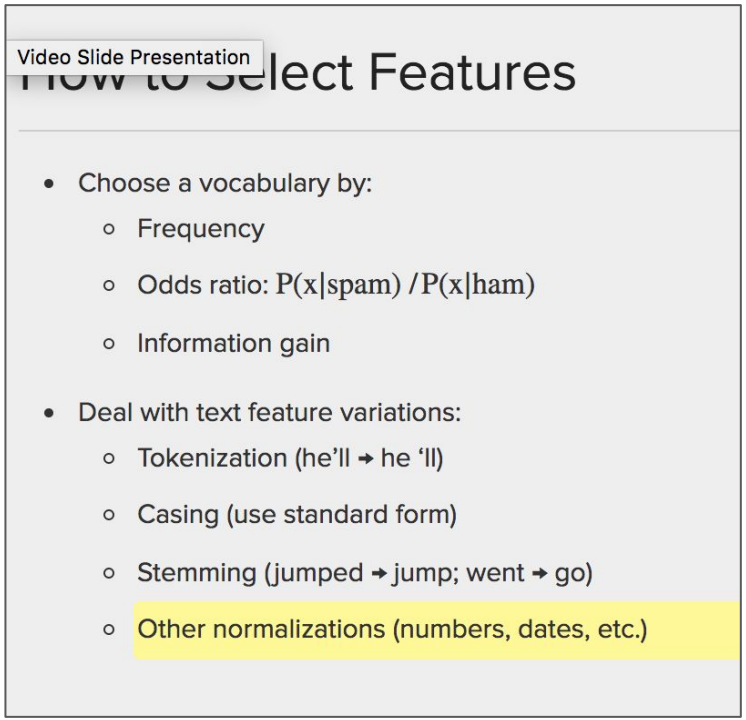
- Update our belief about X, given evidence E.

$$\begin{aligned} P(X|E) &= P(X, E) / P(E) \text{ (apply the definition)} \\ &= P(E, X) / P(E) \text{ (reorder the variables)} \\ &= P(X) P(E|X) / P(E) \text{ (apply the definition)} \end{aligned}$$

- Terminology:
  - **Prior**: $P(X)$
  - **Posterior**: $P(X|E)$
  - **Likelihood**: $P(E|X)$
- Why is Bayes's rule helpful?
  - Often one conditional is easier to come by than the other.

1. Is Naive Bayes a parametric model? Assume a fixed feature set.
2. What is maximum likelihood estimation?
   a. How long does it take to train? Think relative to the size of training data.
   b. Can training be parallelized?
3. How fast is the trained model at making predictions?
   a. How does that compare to KNN?
4. An online learner is one that can update its parameters incrementally given new examples. Why can Naive Bayes be thought of as an 'online model'?

Feature Engineering



1. What might make a feature a 'good' feature?
   a. For house price prediction?
   b. For mortgage approval classification?
   c. Restaurant recommendation?
2. What is feature selection?
   a. Why does it matter with Naive Bayes?

Spam Classification

Naive Bayes for Spam

- Here's our model:
  - Wi is the word at position i in the input.
  - $P(Y|X) \sim P(Y)\,P(W_1|Y)\,P(W_2|Y) \ldots P(W_n|Y)$
- "Bag of Words" (BOW) assumption:
  - Usually, each feature has its own distribution: $P(F_i|Y)$
  - Here, each position has the same distribution: $P(W|Y)$
  - Keeps the number of parameters manageable.

Spam Classification Example

Feature	P(f|spam)	P(f|ham)	Total Spam	Total Ham
(prior)	0.4000	0.6000	−0.92	−0.51
dear	0.0013	0.0009	−7.56	−7.52
sir	0.0023	0.0004	−13.64	−15.35
,	0.0220	0.0241	−17.45	−19.07
first	0.0018	0.0023	−23.77	−25.15
I	0.0062	0.0119	−28.86	−29.57
must	0.0034	0.0028	−34.54	−35.45
solicit	0.0007	0.0002	−41.08	−43.97

- We use log probabilities to prevent underflow.
- $P(\text{spam}|X) = 0.95$
- $P(\text{ham}|X) = 0.05$
- $\text{prediction} = \text{argmax}_y \log P(y) + \sum_i \log P(F_i|Y)$

1. Why do you think Naive Bayes might be so closely associated with text classification? Or put another way, what about NB makes it well suited to working with text based features?
2. Where do the training labels come from in a commercial spam filter?

Generative Modeling

Generative Story for Naive Bayes

- Naive Bayes is a generative model:
  - $P(Y|X) \sim P(Y)P(W_1|Y)P(W_2|Y)...P(W_n|Y)$
- To generate a document:
  - Pick a class spam/ham according to $P(Y)$.
  - Repeat until you have enough words:
    - Pick a word according to $P(W|Y)$.
- Note: Not all models are generative.
  - E.g., logistic regression: not a generative model; it models posterior distribution $P(Y|X)$ directly.

1. What does it mean to be a generative model? Why isn't KNN a generative model?
2. If you generate emails with a NB spam detector, what might those emails look like?

Graham Article

(http://www.paulgraham.com/spam.html)

> I don't know why I avoided trying the statistical approach for so long. I think it was because I got addicted to trying to identify spam features myself, as if I were playing some kind of competitive game with the spammers. (Nonhackers don't often realize this, but most hackers are very competitive.) When I did try statistical analysis, I found immediately that it was much cleverer than I had been. It discovered, of course, that terms like "virtumundo" and "teens" were good indicators of spam. But it also discovered that "per" and "FL" and "ff0000" are good indicators of spam. In fact, "ff0000" (html for bright red) turns out to be as good an indicator of spam as any pornographic term.

1. What are the classes? What are the features? What is the evaluation?
2. What are some of the engineering 'hacks' Graham makes?
3. How fast would his model be to train? Predict? Retrain with addition of one email to training data?

Graham Article

(http://www.paulgraham.com/spam.html)

> Here's a sketch of how I do statistical filtering. I start with one corpus of spam and one of nonspam mail. At the moment each one has about 4000 messages in it. I scan the entire text, including headers and embedded html and javascript, of each message in each corpus. I currently consider alphanumeric characters, dashes, apostrophes, and dollar signs to be part of tokens, and everything else to be a token separator. (There is probably room for improvement here.) I ignore tokens that are all digits, and I also ignore html comments, not even considering them as token separators.
>
> I count the number of times each token (ignoring case, currently) occurs in each corpus. At this stage I end up with two large hash tables, one for each corpus, mapping tokens to number of occurrences.

1. What do stemming and tokenization accomplish from a machine learning perspective?
2. Why do you think Naive Bayes is a popular (baseline) choice for doing text classification?
3. Is spam filtering (necessarily) text classification? (i.e. you build a spam filter without using text as features)?
4. What might be some non-textual features that provide evidence of spam?

Norvig Article

(http://norvig.com/spell-correct.html)

argmax$_{c \in candidates}$ P(c|w)

By Bayes' Theorem this is equivalent to:

argmax$_{c \in candidates}$ P(c) P(w|c) / P(w)

Since P(w) is the same for every possible candidate c, we can factor it out, giving:

argmax$_{c \in candidates}$ P(c) P(w|c)

The four parts of this expression are:

1. **Selection Mechanism**: argmax
   We choose the candidate with the highest combined probability.
2. **Candidate Model**: $c \in candidates$
   This tells us which candidate corrections, c, to consider.
3. **Language Model**: P(c)
   The probability that c appears as a word of English text. For example, occurrences of "the" make up about 7% of English text, so we should have P(*the*) = 0.07.
4. **Error Model**: P(w|c)
   The probability that w would be typed in a text when the author meant c. For example, P(*tehl**the*) is relatively high, but P(*theeexyzl**the*) would be very low.

1. What are the classes? What are the features? What is the evaluation measure?
2. How fast would his model be to train? Predict?
3. What alternatives are there to using edit distance as evidence of P(w|c)?
4. Articles such as this one (and the Graham article) have played a role in popularizing machine learning among engineers. Discuss the pro's and con's of diffusion in this manner?
5. If you were going to train a NB classifier to filter fraudulent rental listings from Craigslist, how might you construct it? How well would you expect it to work?

Final Thoughts?