CPSC 340:
Machine Learning and Data Mining

Generative Models

Admin

- Assignment 0 was due last Wednesday.
  - Because of late days, we have to wait 3 days to post solutions.
  - At that time you will also gain read access to your classmates' work.
    - We voted on this during the first lecture.
    - No one approached me privately with objections.
- Assignment 1 is out.
  - This is a representative assignment w.r.t. length/difficulty/format/style.
- Registration:
  - Keep checking your registration, it could change quickly.
  - As of last night, waitlist was down to 14 people.
- Probability:
  - If you are struggling with probability concepts towards the end of class today, check out the posted notes on probability.

Last Time: Training, Testing, and Validation

- Training step:

  Input: set of 'n' training examples x_i with labels y_i

  Output: a model that maps from arbitrary x_i to a y_i

- Prediction step:

  Input: set of 't' testing examples $\hat{x}_i$ and a model.

  Output: predictions $\hat{y}_i$ for the testing examples.

- What we are interested in is the test error:
  - Error made by prediction step on new data.
- Validation set or cross-validation can be used to estimate test error.

Should you trust them?

- Scenario 1:
  - "I built a model based on the data you gave me."
  - "It classified your data with 98% accuracy."
  - "It should get 98% accuracy on the rest of your data."

- Probably not:
  - They are reporting training error.
  - This might have nothing to do with test error.
  - E.g., they could have fit a very deep decision tree.
- Why 'probably'?
  - If they only tried a few very simple models, the 98% might be reliable.
  - E.g., they only considered decision stumps with simple 1-variable rules.

Should you trust them?

- Scenario 2:
  - "I built a model based on half of the data you gave me."
  - "It classified the other half of the data with 98% accuracy."
  - "It should get 98% accuracy on the rest of your data."

- Probably:
  - They computed the validation error once.
  - This is an unbiased approximation of the test error.
  - Trust them if you believe they didn't violate the golden rule.

Should you trust them?

- Scenario 3:
  - "I built 10 models based on half of the data you gave me."
  - "One of them classified the other half of the data with 98% accuracy."
  - "It should get 98% accuracy on the rest of your data."

- Probably:
  - They computed the validation error a small number of times.
  - Maximizing over these errors is a biased approximation of test error.
  - But they only maximized it over 10 models, so bias is probably small.
  - They probably know about the golden rule.

Should you trust them?

- Scenario 4:
  - "I built 1 billion models based on half of the data you gave me."
  - "One of them classified the other half of the data with 98% accuracy."
  - "It should get 98% accuracy on the rest of your data."

- Probably not:
  - They computed the validation error a huge number of times.
  - Maximizing over these errors is a biased approximation of test error.
  - They tried so many models, one of them is likely to work by chance.
    - This is the "multiple comparisons problem" in statistics
- Why 'probably'?
  - If the 1 billion models were all extremely-simple, 98% might be reliable.

Should you trust them?

- Scenario 5:
  - "I built 1 billion models based on the first third of the data you gave me."
  - "One of them classified the second third of the data with 98% accuracy."
  - "It also classified the last third of the data with 98% accuracy."
  - "It should get 98% accuracy on the rest of your data."

- Probably:
  - They computed the first validation error a huge number of times.
  - But they had a second validation set that they only looked at once.
  - The second validation set gives unbiased test error approximation.
  - This is ideal, as long as they didn't violate golden rule on second set.
  - And assuming you are using IID data in the first place.

The 'Best' Machine Learning Model

- Decision trees are not always most accurate.
- What is the 'best' machine learning model?
- First we need to define generalization error:
  - Test error on new examples (excludes test examples seen during training).
- No free lunch theorem:
  - There is **no** 'best' model achieving the best generalization error for every problem.
  - If model A generalizes better to new data than model B on one dataset, there is another dataset where model B works better.
- This question is like asking which is 'best' among "rock", "paper", and "scissors".

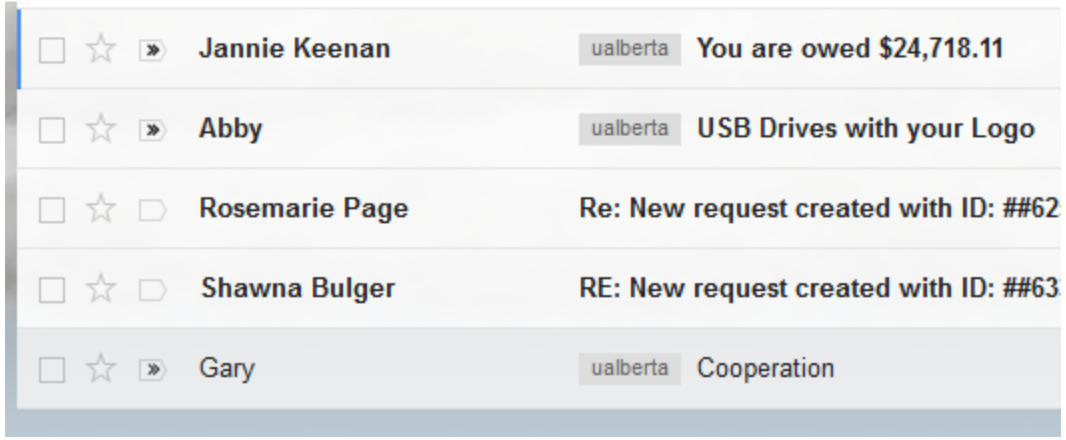
The 'Best' Machine Learning Model

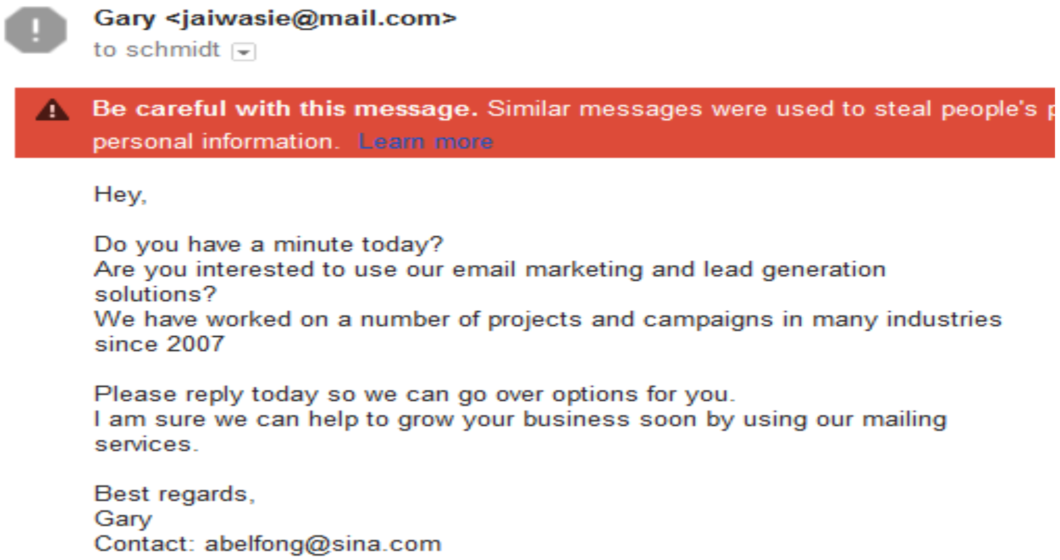
- Implications of the lack of a 'best' model:
  - We need to learn about and try out multiple models.
- So which ones to study in CPSC 340?
  - We'll usually motivate a method by a specific application.
  - But we'll focus on models that are effective in many applications.

- Caveat of no free lunch (NFL) theorem:
  - The world is very structured.
  - Some datasets are more likely than others.
  - Model A really could be better than model B on every real dataset in practice.
- Machine learning research:
  - Large focus on models that are useful across many applications.

Application: E-mail Spam Filtering

- Want a build a system that filters spam e-mails.

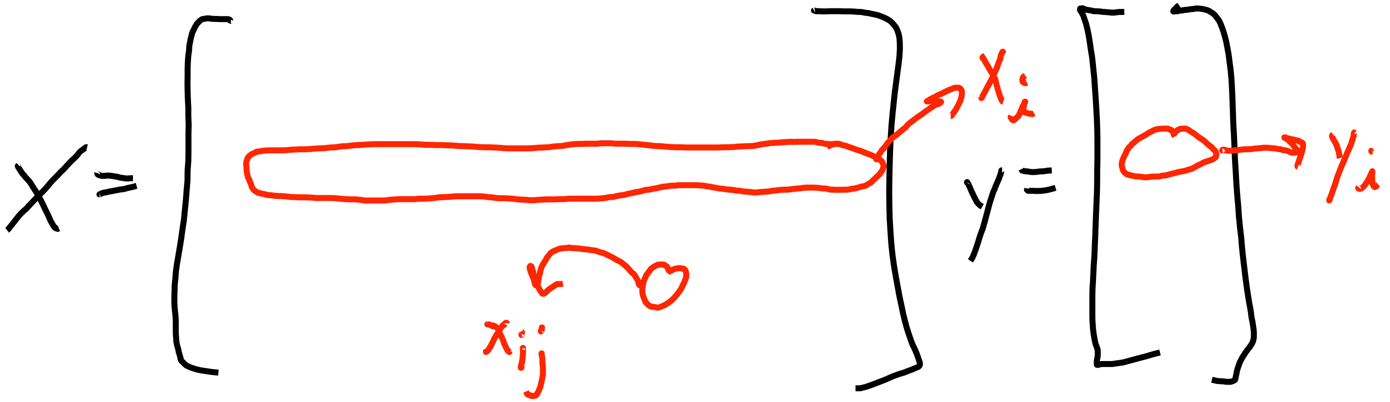




- We have a big collection of e-mails, labeled by users.
- Can we formulate as supervised learning?

First a bit more supervised learning notation

- We have been using the notation ‘X’ and ‘y’ for supervised learning:



- X is matrix of all features, y is vector of all labels.
- Need a way to refer to the features and label of specific object ‘i’.
  - We use y_i for the label of object ‘i’ (element ‘i’ of ‘y’).
  - We use x_i for the features object ‘i’ (row ‘i’ of ‘X’).
  - We use x_{ij} for feature ‘j’ of object ‘i‘.

Feature Representation for Spam

- How do we make label 'y_i' of an individual e-mail?
  - (y_i = 1) means 'spam', (y_i = 0) means 'not spam'.
- How do we construct features 'x_i' for an e-mail?
  - Use bag of words:
    - "hello", "vicodin", "$".
    - "vicodin" feature is 1 if "vicodin" is in the message, and 0 otherwise.
  - Could add phrases:
    - "be your own boss", "you're a winner", "CPSC 340".
  - Could add regular expressions:
    - <recipient>, <sender domain == "mail.com">

Probabilistic Classifiers

- For years, best spam filtering methods used naïve Bayes.
  - Naïve Bayes is a probabilistic classifier based on Bayes rule.
  - It's "naïve" because it makes a strong conditional independence assumption.
  - But it tends to work well with bag of words.

- Probabilistic classifiers model the conditional probability, $p(y_i \mid x_i)$.
  - "If a message has words x_i, what is probability that message is spam?"

- If $p(y_i = \text{'spam'} \mid x_i) > p(y_i = \text{'not spam'} \mid x_i)$, classify as spam.

- Recall our spam filtering setup:
  - y_i: whether or not the e-mail was spam.
  - x_i: the set of words/phrases/expressions in the e-mail.
- To model conditional probability, naïve Bayes uses Bayes rule:

$$p(y_i = \text{"spam"} \mid x_i) = \frac{p(x_i \mid y_i = \text{"spam"})\, p(y_i = \text{"spam"})}{p(x_i)}$$

- Easy part #1: $p(y_i$ = 'spam') is the probability that an e-mail is spam.
  - Count of number of times (y_i = 'spam') divided by number of objects 'n'.

- Recall our spam filtering setup:
  - y_i: whether or not the e-mail was spam.
  - x_i: the set of words/phrases/expressions in the e-mail.
- To model conditional probability, naïve Bayes uses Bayes rule:

$$p(y_i = \text{"spam"} \mid x_i) = \frac{p(x_i \mid y_i = \text{"spam"})\, p(y_i = \text{"spam"})}{p(x_i)}$$

- Easy part #2: We don't need $p(x_i)$.

To test $p(y_i = \text{"spam"} \mid x_i)$ we just need to know if $p(y_i = \text{"spam"} \mid x_i) > p(y_i = \text{"not spam"} \mid x_i)$.

By Bayes rule this is equivalent to $\frac{p(x_i \mid y_i = \text{"spam"})\, p(y_i = \text{"spam"})}{p(x_i)} > \frac{p(x_i \mid y_i = \text{"not spam"})\, p(y_i = \text{"not spam"})}{p(x_i)}$

Denominators are the same so we just test $p(x_i \mid y_i = \text{"spam"})\, p(y_i = \text{"spam"}) > p(x_i \mid y_i = \text{"not spam"})\, p(y_i = \text{"not spam"})$

Generative Classifiers

- The hard part is estimating $p(x_i \mid y_i = \text{'spam'})$:
  - the probability of seeing the words/expressions x_i if the e-mail is spam.
- Classifiers based on Bayes rule are called generative classifier:
  - It needs to know the probability of the features, given the class.
    - How to "generate" features.
  - You need a model that knows what spam messages look like.
    - And a second that knows what non-spam messages look like.
  - This work well with tons of features compared to number of objects.

Generative Models

- Spam filtering methods based on generative models:

$$p(y_i = \text{"spam"} \mid x_i) = \frac{p(x_i \mid y_i = \text{"spam"})\, p(y_i = \text{"spam"})}{p(x_i)}$$

- What do these terms mean?

ALL E-MAILS
(including duplicates)

Generative Models

- Spam filtering methods based on generative models:

$$p(y_i = \text{"spam"} \mid x_i) = \frac{p(x_i \mid y_i = \text{"spam"})\, p(y_i = \text{"spam"})}{p(x_i)}$$

- $p(x_i)$ is probability that a random e-mail has features x_i.

ALL E-MAILS

(including duplicates)

Generative Models

- Spam filtering methods based on generative models:

$$p(y_i = \text{"spam"} \mid x_i) = \frac{p(x_i \mid y_i = \text{"spam"})\,p(y_i = \text{"spam"})}{p(x_i)}$$

- $p(x_i)$ is probability that a random e-mail has features x_i.

ALL E-MAILS

(including duplicates)

$$p(x_i) = \frac{\#\text{ e-mails with features } x_i}{\#\text{ e-mails total}}$$

Generative Models

- Spam filtering methods based on generative models:

$$p(y_i = \text{"spam"} \mid x_i) = \frac{p(x_i \mid y_i = \text{"spam"})\,p(y_i = \text{"spam"})}{p(x_i)}$$

- $p(x_i)$ is probability that a random e-mail has features x_i.

$$p(x_i) = \frac{\#\text{ e-mails with features } x_i}{\#\text{ e-mails total}}$$

Generative Models

- Spam filtering methods based on generative models:

$$p(y_i = \text{"spam"} \mid x_i) = \frac{p(x_i \mid y_i = \text{"spam"})\, p(y_i = \text{"spam"})}{p(x_i)}$$

- $p(x_i)$ is probability that a random e-mail has features x_i.

$$p(x_i) = \frac{\#\text{ e-mails with features } x_i}{\#\text{ e-mails total}}$$

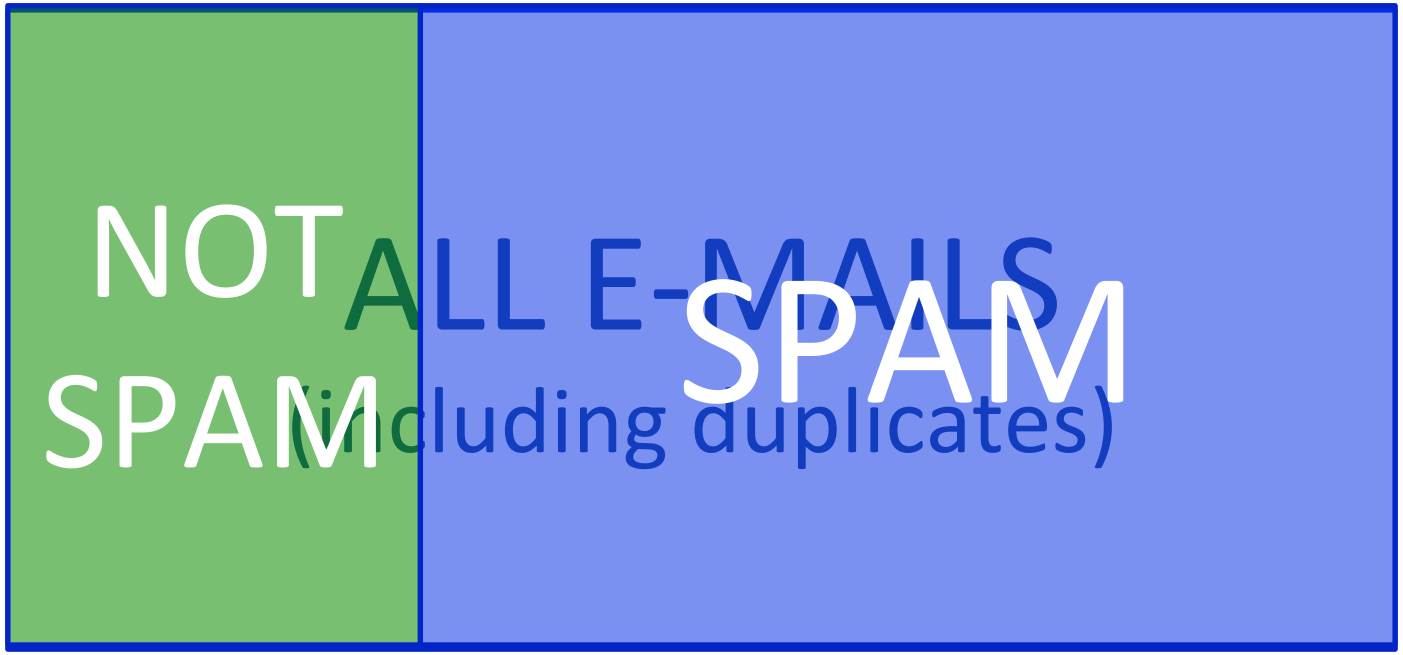
- Hard, but not needed to classify using: $p(y_i = \text{'spam'} \mid x_i) > p(y_i = \text{'not spam'} \mid x_i)$

Generative Models

- Spam filtering methods based on generative models:

$$p(y_i = \text{"spam"} \mid x_i) = \frac{p(x_i \mid y_i = \text{"spam"})\, p(y_i = \text{"spam"})}{p(x_i)}$$

- $p(y_i$ = 'spam') is probability that a random e-mail is spam.



$$p(y_i = \text{"spam"}) = \frac{\#\text{ spam messages}}{\#\text{ total messages}}$$

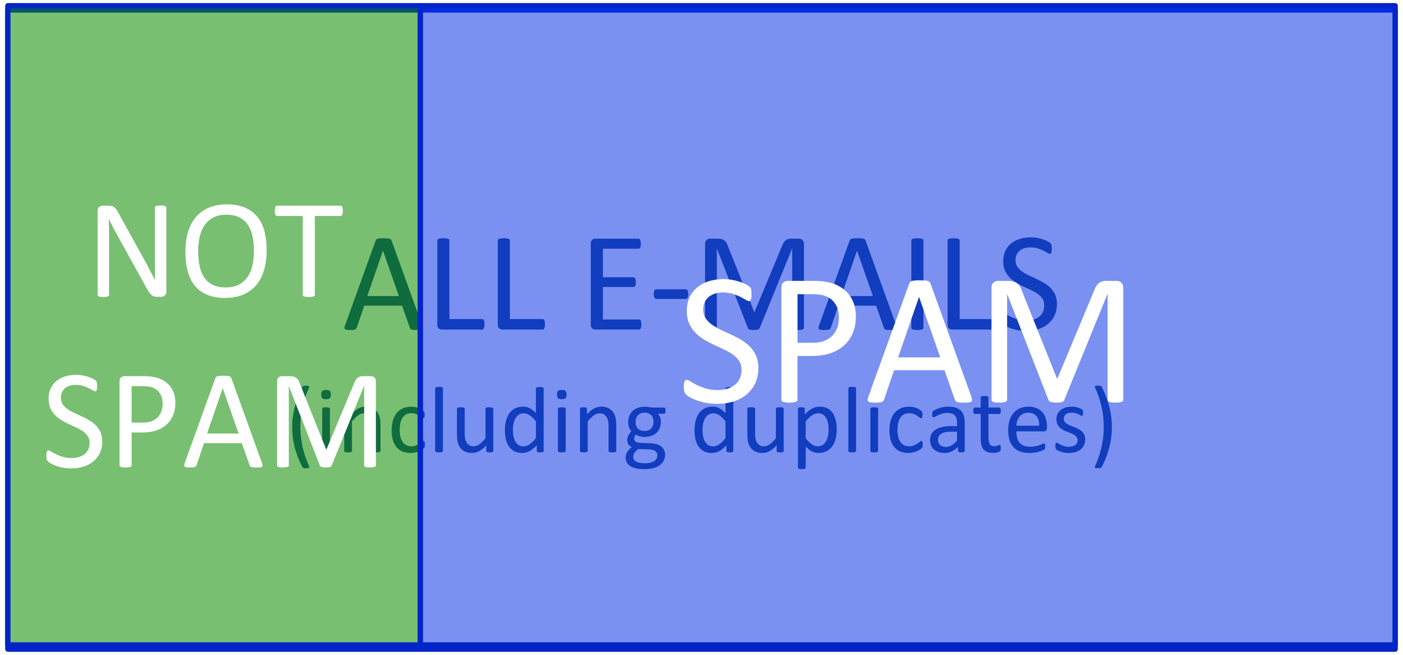
- Hard to compute exactly.
- But is easy to approximate from data:
  - Count (#spam in data)/(#messages)

Generative Models

- Spam filtering methods based on generative models:

$$p(y_i = \text{"spam"} \mid x_i) = \frac{p(x_i \mid y_i = \text{"spam"})\, p(y_i = \text{"spam"})}{p(x_i)}$$

- $p(x_i \mid y_i = \text{'spam'})$ is probability that spam has features x_i.



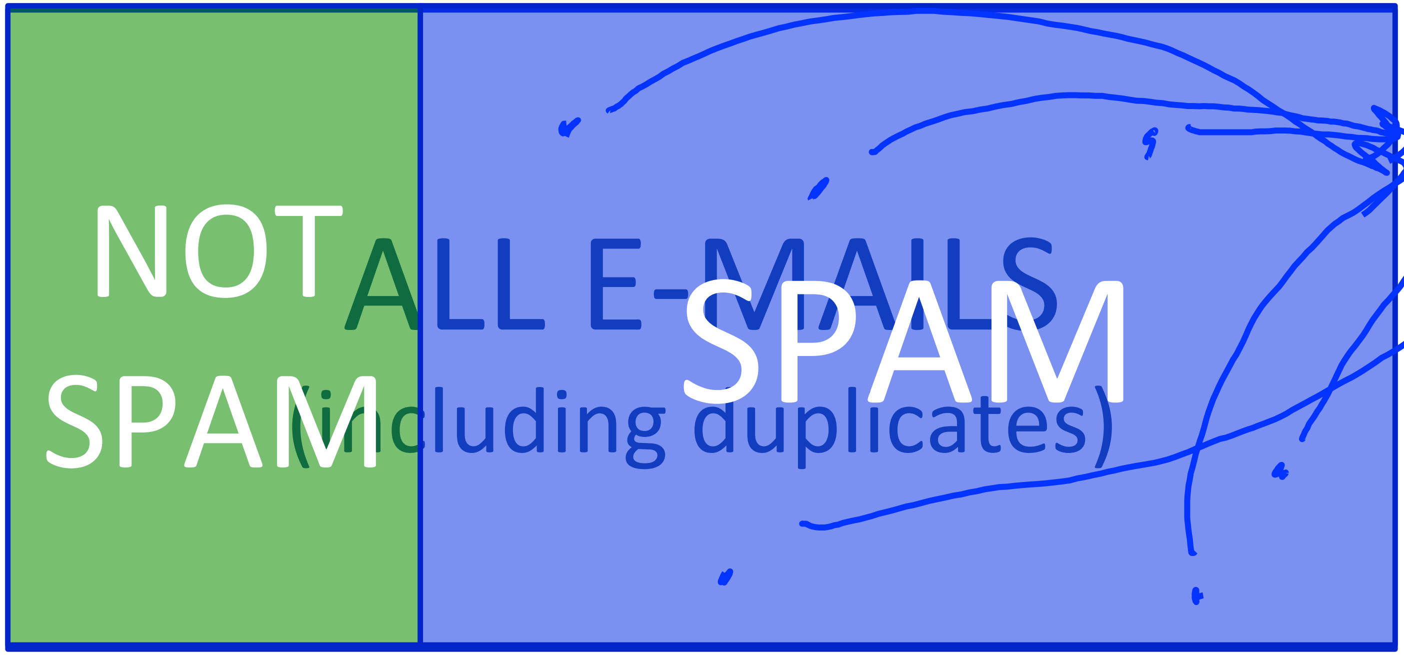
$$p(x_i \mid y_i = \text{"spam"}) = \frac{\#\ \text{spam messages with features } x_i}{\#\ \text{spam messages}}$$

Generative Models

- Spam filtering methods based on generative models:

$$p(y_i = \text{"spam"} \mid x_i) = \frac{p(x_i \mid y_i = \text{"spam"})\, p(y_i = \text{"spam"})}{p(x_i)}$$

- $p(x_i \mid y_i = \text{'spam'})$ is probability that spam has features x_i.



$$p(x_i \mid y_i = \text{"spam"}) = \frac{\#\ \text{spam messages with features } x_i}{\#\ \text{spam messages}}$$

- Very hard to estimate:
  - Too many possible x_i.

Naïve Bayes

- How the naïve Bayes model deals with the hard terms:

$$p(\text{spam} \mid \text{hello, vicodin, CPSC 340}) = \frac{p(\text{hello, vicodin, CPSC 340} \mid \text{spam})\, p(\text{spam})}{p(\text{hello, vicodin, CPSC 340})} \quad \text{(Bayes rule)}$$

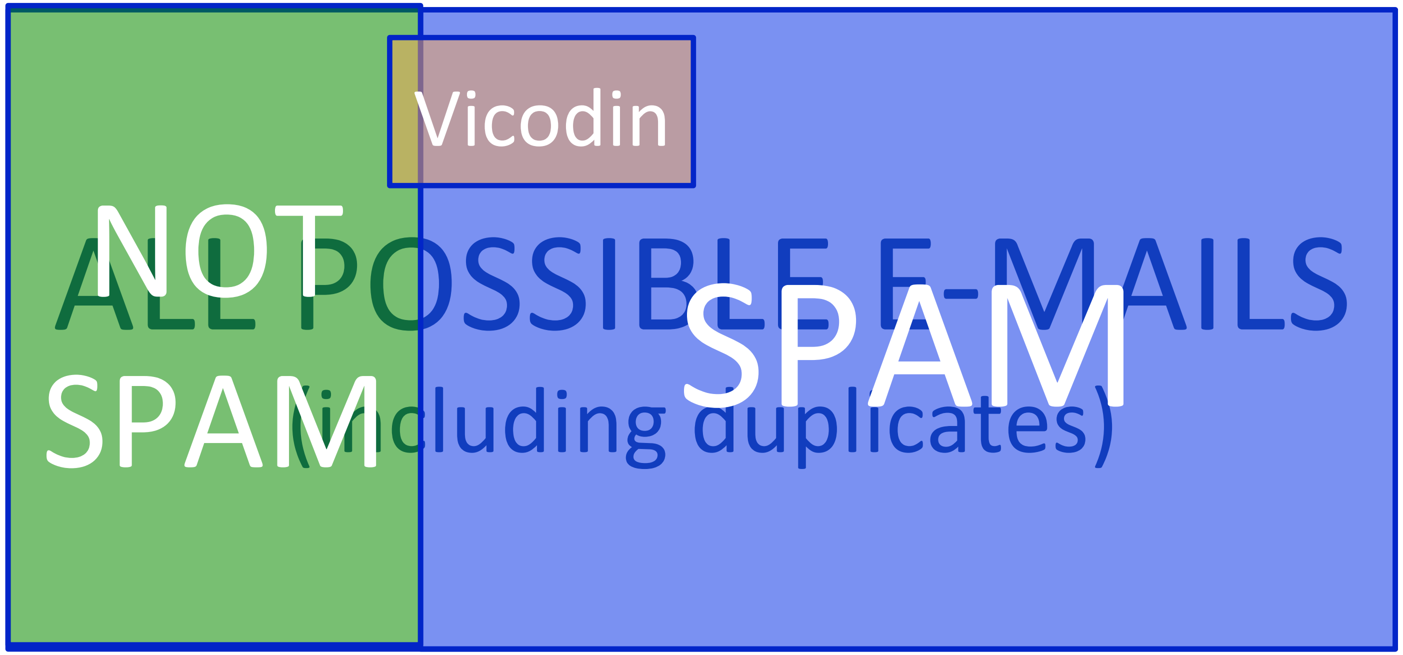
"equal up to constant not depending on spam" $\propto \underbrace{p(\text{hello, vicodin, CPSC 340} \mid \text{spam})}_{\text{HARD}} \underbrace{p(\text{spam})}_{\text{easy}}$

(naive Bayes assumption) $\approx \underbrace{p(\text{hello} \mid \text{spam})}_{\text{easy}} \underbrace{p(\text{vicodin} \mid \text{spam})}_{\text{easy}} \underbrace{p(\text{CPSC 340} \mid \text{spam})}_{\text{easy}} \underbrace{p(\text{spam})}_{\text{easy}}$

- Now only need easy quantities like p('vicodin' = 1| y_i = 'spam').

Naïve Bayes Models

- p(vicodin = 1 | spam = 1) is probability of seeing ‘vicodin’ in spam.



$$p(\text{vicodin}=1 \mid \text{spam}=1) = \frac{\#\text{ spam messages w/ vicodin}}{\#\text{ spam messages}}$$

- Easy to estimate:
  - #(spam w/ Vicodin)/#spam
- “Maximum likelihood estimate”

- In naïve Bayes: assume features are independent given label.
  - “Once you know it’s spam, there is no dependency between features.”
  - Not true, but sometimes a good approximation.

Naïve Bayes

- Naïve Bayes more formally:

$$p(y_i | x_i) = \frac{p(x_i | y_i) p(y_i)}{p(x_i)}$$

$$\propto p(x_i | y_i) p(y_i)$$

$$\approx \prod_{j=1}^{d} \left[p(x_{ij} | y_i) \right] p(y_i)$$

  – Assumption: all x_i are conditionally independent given y_i.

Independence of Random Variables

- Events A and B are independent if $p(A,B) = p(A)p(B)$.
  - Equivalently: $p(A|B) = p(A)$.
  - "Knowing B happened tells you nothing about A".
  - We use the notation:

$$A \perp B$$

- Random variables are independent if $p(x,y) = p(x)p(y)$ for all x and y.
  - Flipping two coins:

$$p(C_1 = \text{'heads'}, C_2 = \text{'heads'}) = p(C_1 = \text{'heads'})p(C_2 = \text{'heads'}).$$
$$p(C_1 = \text{'tails'}, C_2 = \text{'heads'}) = p(C_1 = \text{'tails'})p(C_2 = \text{'heads'}).$$
$$\ldots$$

Conditional Independence

- Example: food poisoning
  - If food was bad, each person independently gets sick with probability 50%
  - Unconditionally, me getting and and you getting sick are NOT independent
    - If I got sick, that makes me think the food was bad, which makes it more likely that you will get sick also. So knowing my situation influences my beliefs about yours.
  - But, conditioned on knowing the food was bad (or not bad), my sickness and your sickness are independent.
- Definition: A and B are conditionally independent given C if

$$p(A, B \mid C) = p(A \mid C)p(B \mid C).$$

  - Equivalently: $p(A \mid B, C) = p(A \mid C)$.
  - "Knowing C happened, also knowing B happened says nothing about A".
  - We use the notation: $A \perp B \mid C$

Naïve Bayes for any number of classes

- Let c be a class label in $\{c_1, c_2, \ldots\}$
  - In the spam example, we only had 2 classes (spam and not spam)
- Let i be a training example's index, j a feature index, k a feature value

Training:

1. Set n_c to the number of times $(y^i = c)$.
2. Estimate $p(y=c)$ as $\frac{n_c}{n}$.
3. Set n_{cjk} as the number of times $(y_i = c, X_{ij} = k)$
4. Estimate $p(x_i = k \mid y = c) = \frac{p(x_i = k, y = c)}{p(y = c)}$ as $\frac{\frac{n_{cjk}}{n}}{\frac{n_c}{n}} = \frac{n_{cjk}}{n_c}$.

Naïve Bayes for any number of classes

Prediction:

Given a new example x_i we want to find the 'c' maximizing $p(x_i | y_i)$.

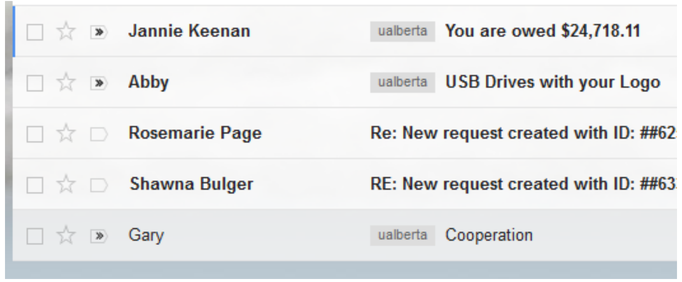
Under the naive Bayes assumption we thus maximize

$$p(y=c \mid x_i) \propto \prod_{j=1}^{d} \left[p(x_{ij} \mid y=c) \right] p(y=c)$$

- Note that these terms do not add up to 1 because we dropped the denominator $p(x_i)$.

Application: E-mail Spam Filtering

- Want a build a system that filters spam e-mails:



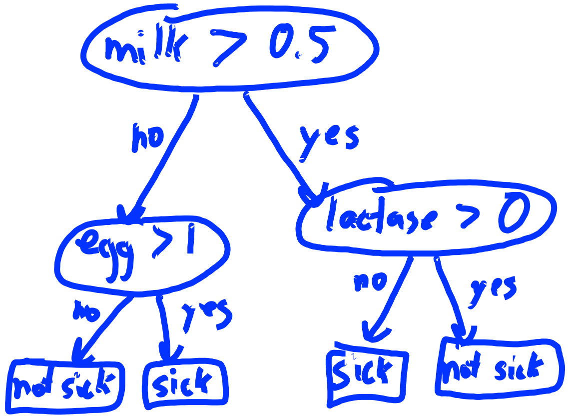
- We formulated as supervised learning:
  - (y_i = 1) if e-mail 'i' is spam, (y_i = 0) if e-mail is not spam.
  - (xij = 1) if word/phrase 'j' is in e-mail 'i', (xij = 0) if it is not.

$	Hi	CPSC	340	Vicodin	Offer	…
1	1	0	0	1	0	…
0	0	0	0	1	1	…
0	1	1	1	0	0	…
…	…	…	…	…	…	…

Spam?
1
1
0
…

Decision Trees vs. Naïve Bayes

- Decision trees:



1. Sequence of rules based on 1 feature.
2. Training: 1 pass over data per depth.
3. Hard to find optimal tree.
4. Testing: just look at features in rules.
5. New data: might need to change tree.
6. Accuracy: good if simple rules work.

- Naïve Bayes:

$$p(\text{sick} \mid \text{milk}, \text{egg}, \text{lactase})$$
$$\approx p(\text{milk} \mid \text{sick})\, p(\text{egg} \mid \text{sick})\, p(\text{lactose} \mid \text{sick})\, p(\text{sick})$$

1. Simultaneously combine all features.
2. Training: 1 pass over data to count.
3. Easy to find optimal probabilities.
4. Testing: look at all features.
5. New data: just update counts.
6. Accuracy: good if features almost independent given label.

Naïve Bayes Issues

1. Do we need to store the full bag of words 0/1 variables?
   - No: only need list of non-zero features for each e-mail.
     - Could use a sparse matrix representation.
2. Problem with maximum likelihood estimate (MLE):
   - MLE of p('lactase' = 1| 'spam') is (#spam messages with 'lactase')/#spam.
   - If you have no spam messages with lactase:
     - p('lactase' | 'spam') = 0, and message automatically gets through filter.
   - Fix: imagine we saw/not-saw each word in spam/not-spam messages:
     - "Laplace smoothing": assume some "pseudo-counts" for each feature/label.
     - for binary features: replace n_{cjk}/n_c with $(n_{cjk} + 1)/(n_c + 2)$.
       - a generalization is $(n_{cjk} + \beta)/(n_c + 2\beta)$ for some constant β.
     - If X_{ij} can take 'm' values, you would $(n_{cjk} + \beta)/(n_c + m\beta)$

Naïve Bayes Issues

3. During the prediction, the probability can underflow:

$$p(y=c \mid x_i) \propto \prod_{j=1}^{d} \left[p(x_{ij} \mid y=c) \right] p(y=c)$$

All these are < 1 so the product gets very small.

- Standard fix is to (equivalently) maximize the logarithm of the probability:
  - Logarithm turns multiplication of small numbers into addition of small numbers.
  - Logarithm is monotonic, so it doesn't change location of the maximum (maximizer).
  - See CPSC 302/303 for more on underflow and floating point issues.

Decision Theory

- Spam classification example
  – Are we equally concerned about spam vs. not spam?
- True positives, false positives, false negatives, false negatives:

Predict / True	True ‘spam’	True ‘not spam’
Predict ‘spam’	True Positive	False Positive
Predict ‘not spam’	False Negative	True Negative

- The costs mistakes might be different:
  – Letting a spam message through (false negative) is not a big deal.
  – Filtering a not spam (false positive) message will make users mad.

Decision Theory

- We can give a cost to each scenario, such as:

Predict / True	True ‘spam’	True ‘not spam’
Predict ‘spam’	0	100
Predict ‘not spam’	10	0

- Instead of assigning to most likely classify, minimize expected cost:

$$E[C(\hat{y}_i = \text{spam})] = p(y_i = \text{spam} \mid x_i)\, C(\hat{y}_i = \text{spam}, y_i = \text{spam}) + p(y_i = \text{not spam} \mid x_i)\, \underbrace{C(\hat{y}_i = \text{spam}, y_i = \text{not spam})}_{\text{"cost of predicting spam when e-mail is not spam"}}$$

- Even if $p(\text{spam} \mid x_i) > p(\text{not spam} \mid x_i)$,
  - Might still classify as “not spam”,
    if $E[C(\text{yhat}_i = \text{spam})] > E[C(\text{yhat}_i = \text{not spam})]$.

Decision Theory and Darts

- Post on decision theory in “darts”:
  - http://www.datagenetics.com/blog/january12012/index.html

- If you are very accurate, aim for the high-scoring regions.
- If you are very inaccurate, aim for the middle.
- Decision theory gives you the best strategy for other accuracies.

Summary

- No free lunch theorem: there is no "best" ML model.
- Joint probability: probability of A and B happening.
- Conditional probability: probability of A if we know B happened.
- Generative classifiers: build a probability of seeing the features.
  - Naïve Bayes uses conditional independence assumptions to make estimation practical.
- Decision theory allows us to consider costs of predictions.

- Next time:
  - A "best" machine learning model as 'n' goes to ∞.

- All the remaining slides are “bonus”.
- We may go through them briefly, if time permits.

Generative Classifiers

- But does it need to know language to model $p(x_i \mid y_i)$???
- To fit generative models, usually make BIG assumptions:
  - Gaussian discriminant analysis (GDA):
    - Assume that $p(x_i \mid y_i)$ follows a multivariate normal distribution.
  - Naïve Bayes (NB):
    - Assume that each variables in x_i is independent of the others in x_i given y_i.

Bonus Slide: Avoiding Underflow

- During the prediction, the probability can underflow:

$$p(y=c \mid x_i) \propto \underbrace{\prod_{j=1}^{d}\left[p(x_{ij} \mid y=c)\right] p(y=c)}$$

All these are < 1 so the product gets very small

- Standard fix is to (equivalently) maximize the logarithm of the probability:

Rember that $\log(ab) = \log(a) + \log(b)$ so $\log(\prod a_i) = \sum_i \log(a_i)$

Since log is monotonic the 'c' maximizing $p(y=c \mid x_i)$ also maximizes

$$\log p(y_i = c \mid x_i) = \sum_{j=1}^{d}\left[p(x_{ij} \mid y=c)\right] + p(y=c) + \underbrace{\text{constant}}_{\text{which is the same for all 'c'}}$$

Bonus Slide: $p(x_i)$ under naïve Bayes

- Generative models don't need $p(x_i)$ to make decisions.
- However, it's easy to calculate under the naïve Bayes assumption:

$$p(x_i) = \sum_{c=1}^{K} p(x_i, y=c) \quad \text{(marginalization rule)}$$

$$= \sum_{c=1}^{K} p(x_i \mid y=c)\, p(y=c) \quad \text{(product rule)}$$

$$= \sum_{c=1}^{K} \underbrace{\left[\prod_{j=1}^{d} p(x_{ij} \mid y=c)\right] p(y=c)}_{\text{These are the quantities we compute during training.}} \quad \text{(naive Bayes assumption)}$$

Bonus Slide: Less-Naïve Bayes

- Given features {x1,x2,x3,…,xd}, naïve Bayes approximates p(y|x) as:

$$p(y \mid x_1, x_2, \dots, x_d) \propto p(y)\, p(x_1, x_2, \dots, x_d \mid y)$$

product rule applied repeatedly

$$= p(y)\, p(x_1 \mid y)\, p(x_2 \mid x_1, y)\, p(x_3 \mid x_2, x_1, y) \cdots p(x_d \mid x_1, x_2, \dots, x_{d-1}, y)$$

$$\approx p(y)\, p(x_1 \mid y)\, p(x_2 \mid y)\, p(x_3 \mid y) \cdots p(x_d \mid y) \quad \text{(naive Bayes assumption)}$$

- The assumption is very strong, and there are "less naïve" versions:
  - Assume independence of all variables except up to 'k' largest 'j' where j < i.
    - E.g., naïve Bayes has k=0 and with k=2 we would have:

$$\approx p(y)\, p(x_1 \mid y)\, p(x_2 \mid x_1, y)\, p(x_3 \mid x_2, x_1, y)\, p(x_4 \mid x_3, x_2, y) \cdots p(x_d \mid x_{d-2}, x_{d-1}, y)$$

    - Fewer independence assumptions so more flexible, but hard to estimate for large 'k'.
  - Another practical variation is "tree-augmented" naïve Bayes.