CPSC 340: Machine Learning and Data Mining

Decision Theory and

Non-Parametric Models

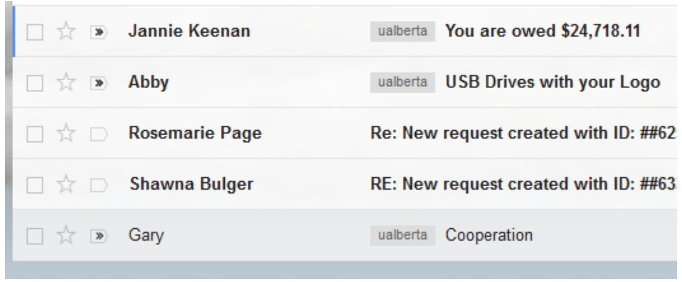
September 18, 2015

Admin

- Assignment 2 out today, due Friday of next week, start early!
- No tutorials today, there will be office hours tomorrow.
- Course drop deadline tomorrow.

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
…

Generative Models

- We considered 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

- We considered 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

- We considered 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

- We considered 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

- We considered 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)

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

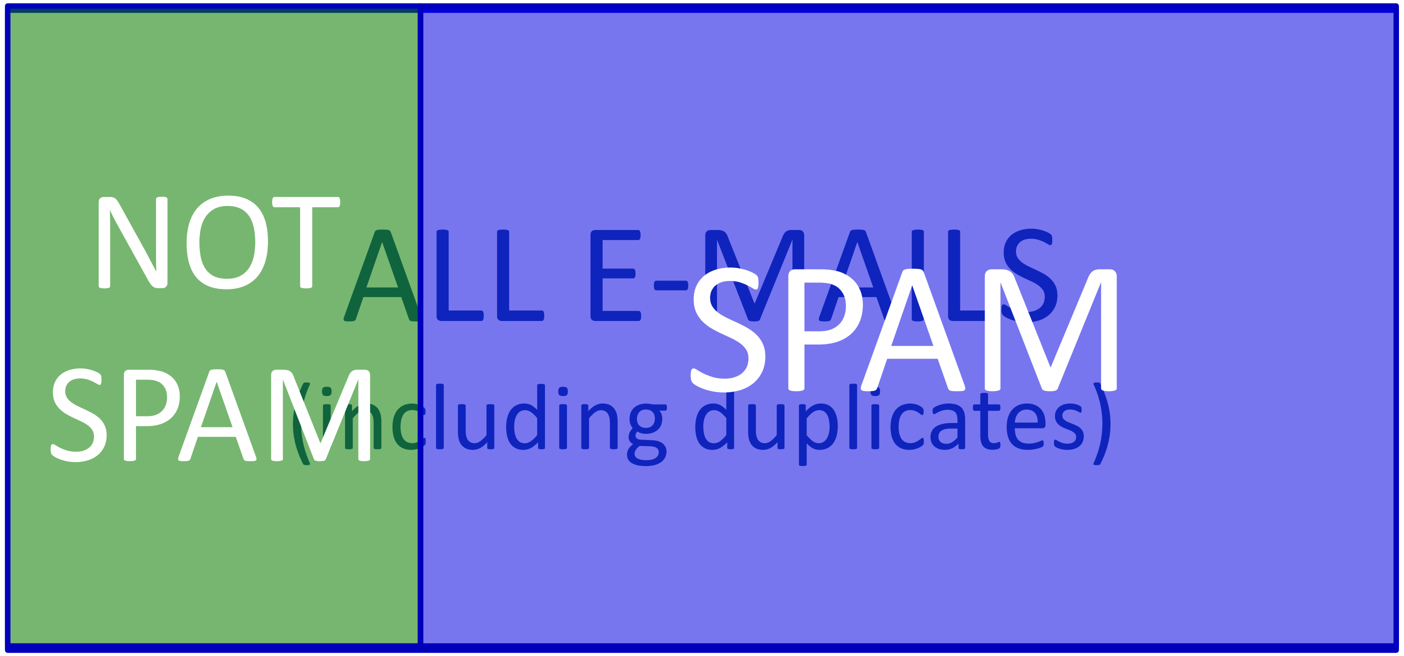
- Hard, but not needed to classify using:
  $p(y_i = \text{'spam'} \mid x_i) > p(y_i = \text{'not spam'} \mid x_i)$

Generative Models

- We considered 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.



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

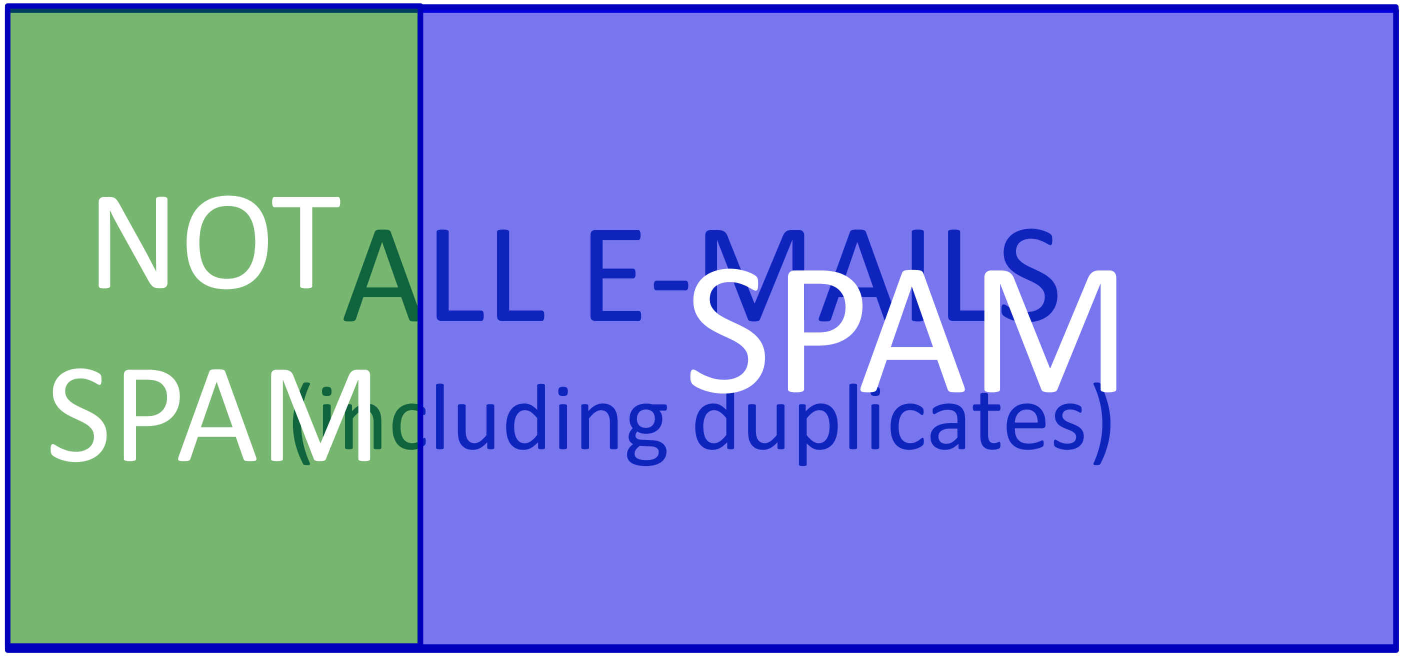
- Hard to compute exactly.
- But is easy to approximate from data:
  - Count (#spam in data)/(#messages)

Generative Models

- We considered 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.



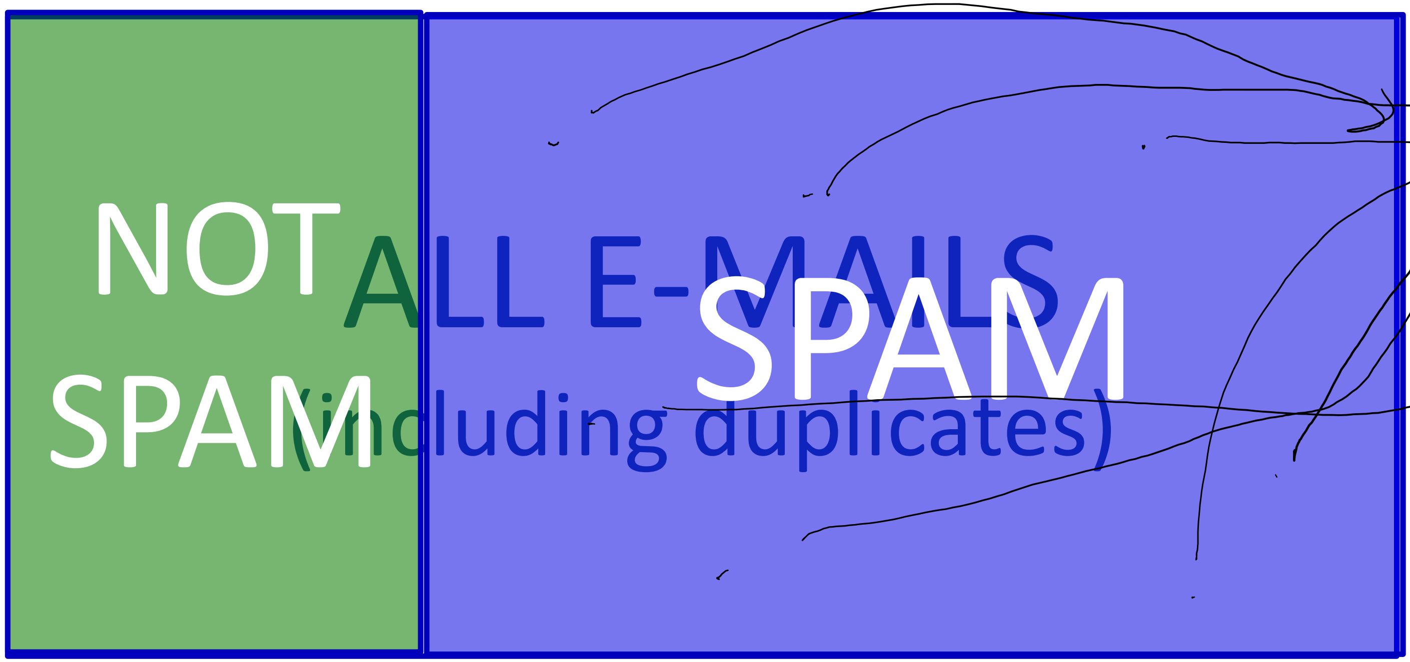
$$p(x_i \mid y_i = \text{'spam'}) = \frac{\#\text{spam messages with features } x_i}{\#\text{spam messages}}$$

Generative Models

- We considered 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.



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

- Hard to compute.

Naïve Bayes

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

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

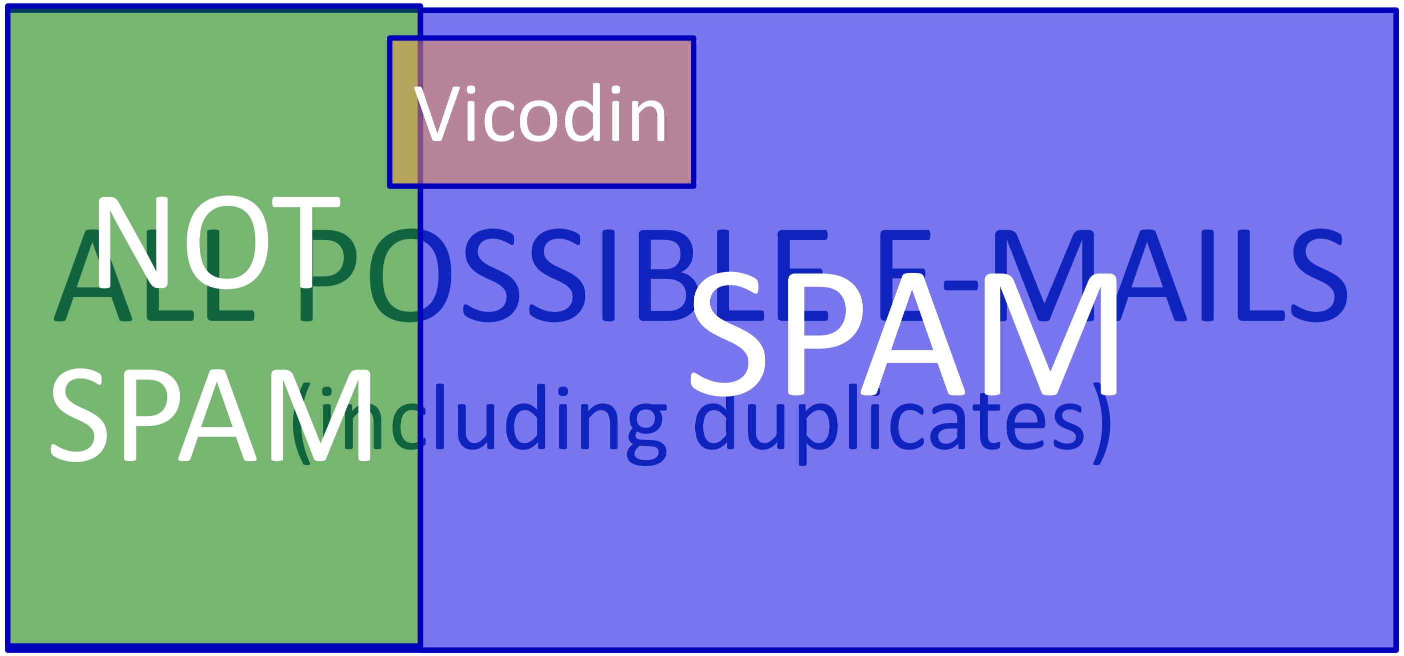
"equal up to constant" $$\propto p(\text{hello}, \text{vicodin}, \text{CPSC 340} \mid \text{spam})\, p(\text{spam})$$

$$\approx p(\text{hello} \mid \text{spam})\, p(\text{vicodin} \mid \text{spam})\, p(\text{CPSC 340} \mid \text{spam})\, p(\text{spam})$$

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

Naïve Bayes Models

- p(vicodin = 1 | spam) is probability of seeing 'vicodin' in spam message.



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

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

Naïve Bayes

- Naïve Bayes more formally:

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

$$\propto p(x_i \mid y_i)\, p(y_i)$$

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

  - Assumption: given y_i, all x_i are conditionally independent of each other.

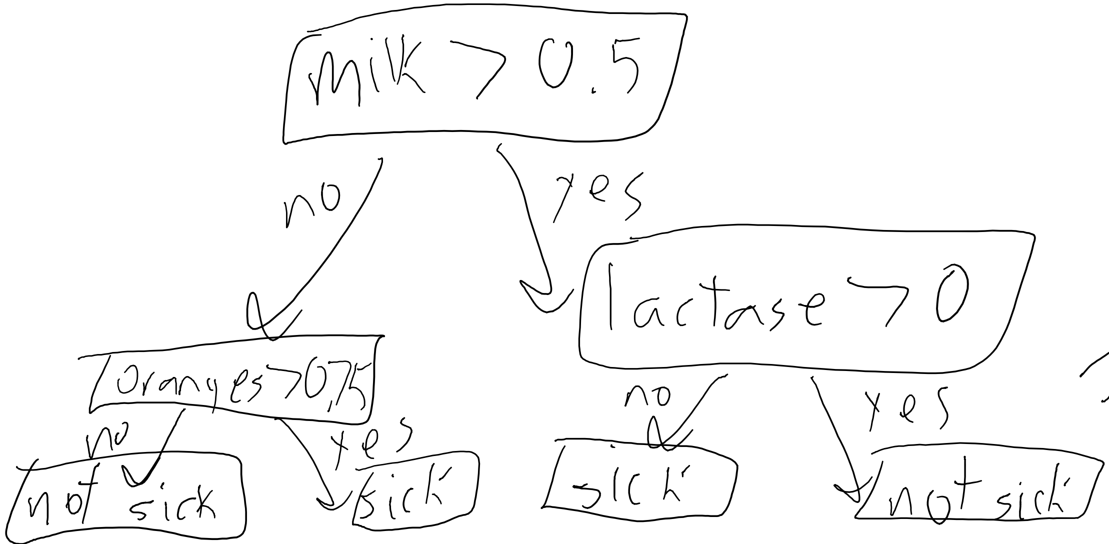
Conditional Independence

- A and B are conditionally independent given C if
  $$p(A, B \mid C) = p(A \mid C)p(B \mid C).$$
  - Equivalently: $p(A \mid B, C) = p(A \mid C)$. or $p(B \mid A, C) = p(B \mid C)$
  - "Knowing C happened, also knowing B happened says nothing about A".
  - Example: $p(\text{Pizza} \mid D_1, \text{Survive}) = p(\text{Pizza} \mid \text{Survive})$.
  - Knowing you survived, dice 1 gives no information about chance of pizza.
  - We use the notation: $A \perp B \mid C$ $\quad \text{Pizza} \perp D_1 \mid \text{Survive}$

- Semantics of $p(A, B \mid C, D)$:
  - "probability of A and B happening, if we know that C and D happened".

Decision Trees vs. Naïve Bayes

- Decision trees:
  - Sequence of rules based on 1 feature.
  - Training: 1 pass over data per depth.
  - Hard to find optimal tree.
  - Testing: just look at features in rules.
  - Accuracy: good if simple rules work.



- Naïve Bayes:
  - Simultaneously combine all features.
  - Training: 1 pass over data.
  - Easy to find optimal probabilities.
  - Testing: look at all features.
  - Accuracy: good if features almost independent given label.

$$p(\text{sick} \mid \text{milk}, \text{oranges}, \text{lactase})$$
$$\propto p(\text{milk}, \text{oranges}, \text{lactase} \mid \text{sick})\, p(\text{sick})$$
$$\approx p(\text{milk} \mid \text{sick})\, p(\text{oranges} \mid \text{sick})\, p(\text{lactase} \mid \text{sick})\, p(\text{sick})$$

Naïve Bayes Issues

1. Do we need to store the full bag of words representation?
   - No: only need list of non-zero features for each e-mail.
   - We use a sparse matrix in Assignments 1 and 2.
2. Problem with maximum likelihood estimate (MLE):
   - MLE of p('lactase' = 1| 'spam') is (#spam messages with 'lactase')/#spam.
   - If you've never seen 'lactase' in a spam message then:
     - p('lactase' | 'spam') = 0, and message automatically gets through filter.
   - Fix: imagine we saw/not-saw each word in spam/not-spam messages:
     - Estimate p(<word> | 'spam') by (1 + count(spam with <word>)/(2 + #spam).
     - We might use parameter 'c' instead of '1', and '2c' instead of '2'.
3. Are we equally concerned about spam vs. not spam?

Decision Theory

- 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 of false positives vs. false negatives 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'	TP: 0	FP: 100
Predict 'not spam'	FN: 10	TN: 0

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

$$E[C(\hat{y}_i = \text{spam})] = p(y_i = \text{spam} \mid x_i)\, C(\hat{y}_i = \text{spam}, y_i = \text{spam})$$
$$+ p(y_i = \text{not spam} \mid x_i)\, C(\hat{y}_i = \text{spam}, y_i = \text{not spam})$$

- Might classify as 'not spam' even if $p(\text{spam} \mid x_i) > p(\text{not spam} \mid x_i)$,

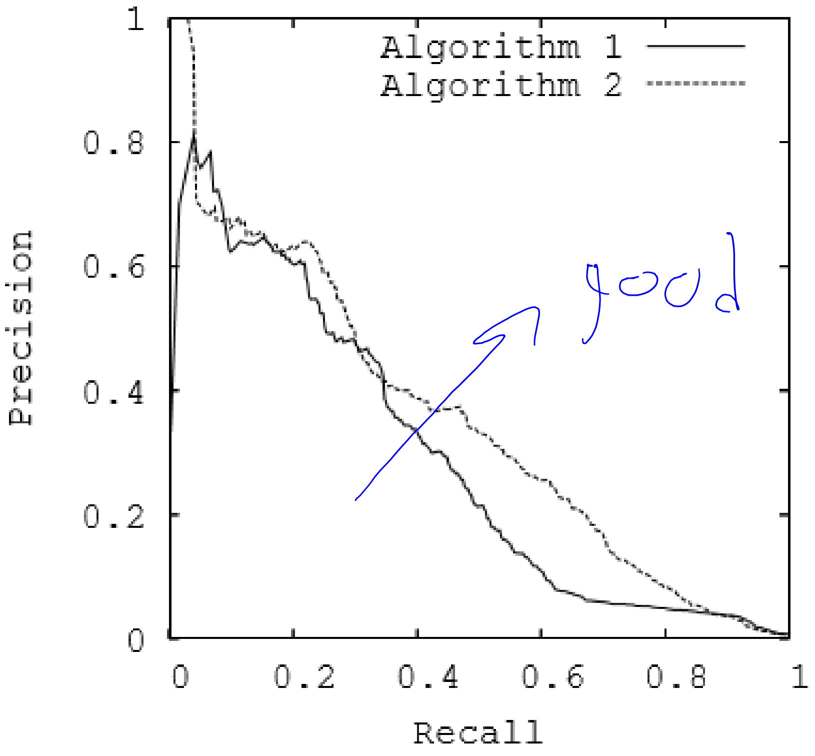
  if $E[C(\text{yhat}_i = \text{spam})] > E[C(\text{yhat}_i = \text{not spam})]$.

Other Performance Measures

- Classification error might be wrong measure:
  - Use weighted classification error if have different costs.
  - Might want to use things like Jaccard measure.
- Often, we report precision and recall (want both to be high):
  - Precision: “if I classify as spam, what is the probability it actually is spam?”
    - Precision = TP/(TP + FP).
    - High precision means the filtered messages are likely to really be spam.
  - Recall: “if a message is spam, what is probability it is classified as spam?”
    - Recall = TP/(TP + FN)
    - High recall means that most spam messages are filtered.

Precision-Recall Curve

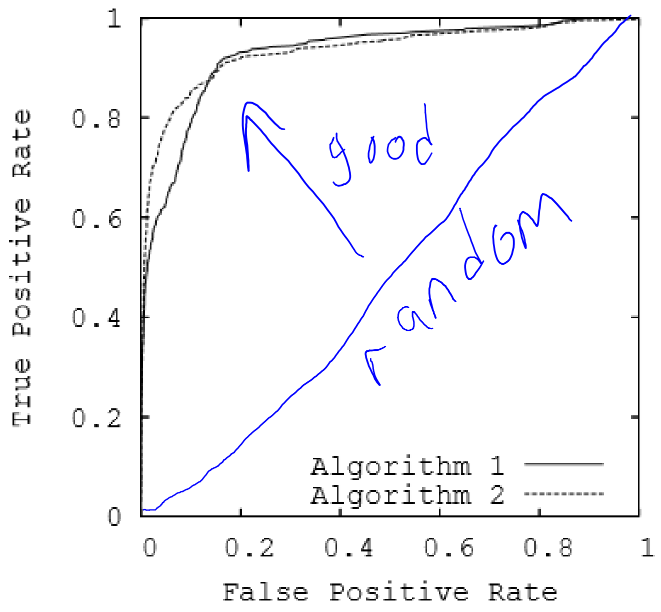
- Consider the rule $p(y_i = \text{'spam'} \mid x_i) > t$, for threshold 't'.
- Precision-recall (PR) curve plots precision vs. recall as 't' varies.



http://pages.cs.wisc.edu/~jdavis/davisgoadrichcamera2.pdf

ROC Curve

- Receiver operating characteristic (ROC) curve:
  - Plot true positive rate (recall) vs. false positive rate (FP/FP+TN).
    (negative examples classified as positive)



  - Diagonal is random, perfect classifier would be in upper left.
  - Sometimes papers report area under curve (AUC).

Parametric vs. Non-Parametric Methods

Parametric vs. Non-Parametric

- Decision trees and naïve Bayes are often not very accurate.
  - Rules or independence assumptions might not make sense in application.
  - They are also parametric methods:
    - There are a fixed number of “parameters” in the model (e.g., number of rules).
    - As you get more data, you can estimate them more accurately.
    - But at some point, more data doesn’t help because model is too simple.
    - E.g., depth-3 decision trees can’t model most distributions.
- Non-parametric models:
  - Number of parameters grows with the number of training examples.
  - Model gets more complicated as you get more data.
  - E.g., decision tree whose depth *grows with the number of examples*.

K-Nearest Neighbours (KNN)

- Classical non-parametric classifier is k-nearest neighbours.
- Based on an intuitive idea:
  - Objects with similar features are likely to have similar labels.
- K-nearest neighbours algorithm for classifying a test example ‘x‘:
  - Find ‘k’ values of x_i that are most similar to x.
  - Find the ‘k’ corresponding labels y_i.
  - Classify using the mode of the y_i.
- “Lazy” learning: there is no actual “training” phase (just store data).
- Number of “parameters” is proportional to data size.

How to Define 'Nearest'?

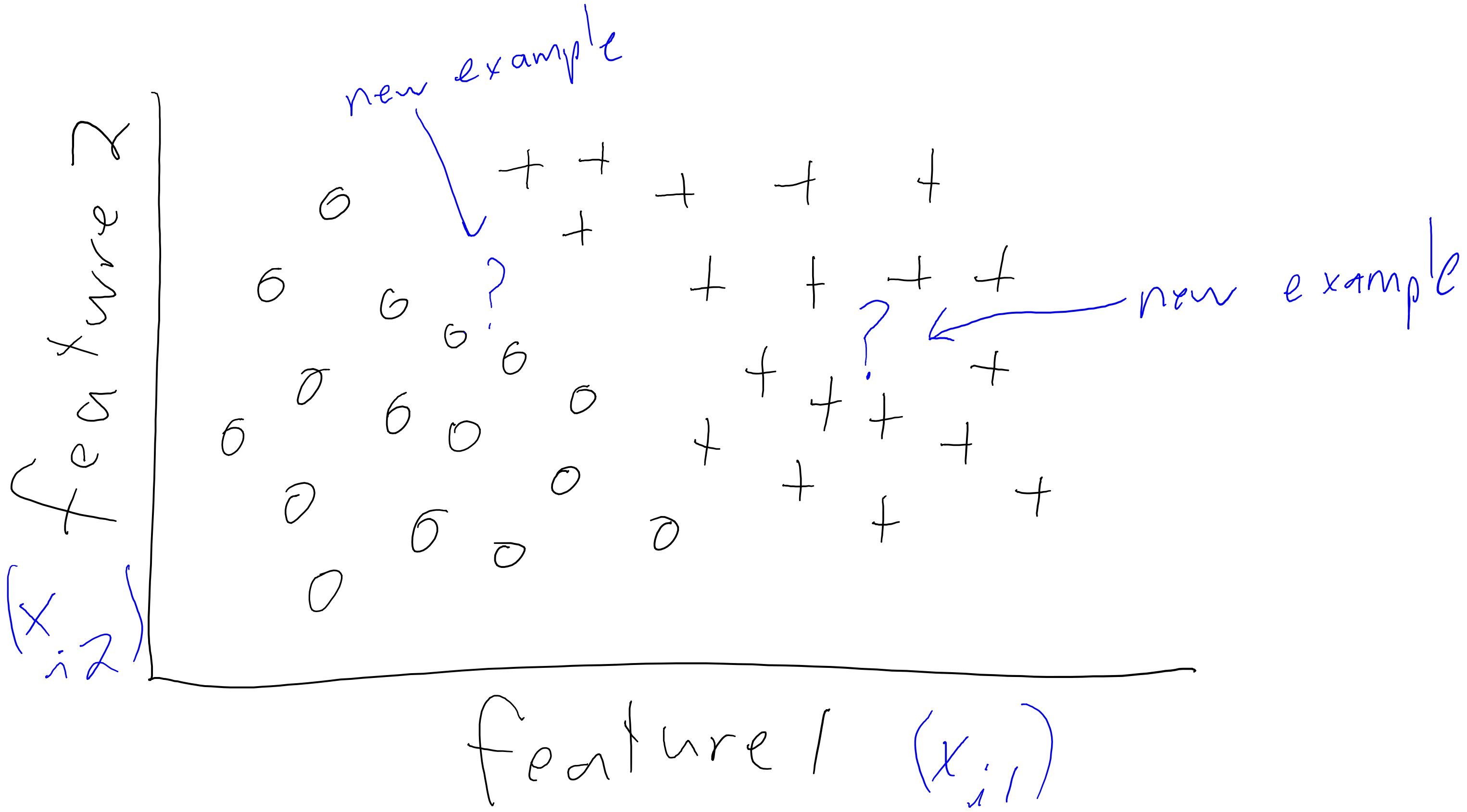
- There are many possible notions of similarity between x_i and x_j.
- Most common is Euclidean distance:

$$D(x_1, x_2) = \sqrt{\sum_{j=1}^{d} (x_{1j} - x_{2j})^2}$$

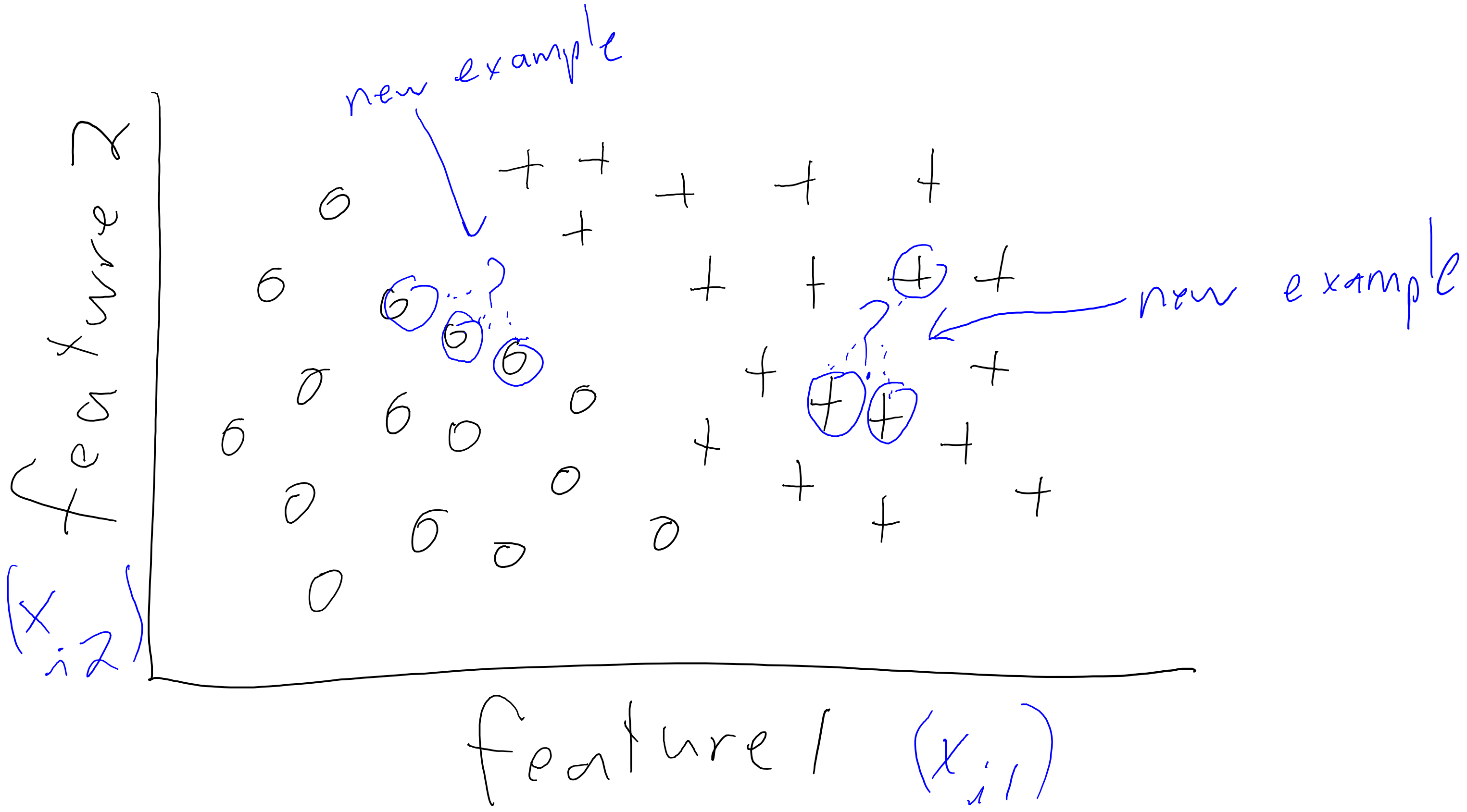
- Other possibilities:
  - L_1 distance: $D(x_1, x_2) = \sum_{j=1}^{d} |x_{1j} - x_{2j}|$
  - Jaccard similarity (binary): $D(x_1, x_2) = \frac{x_1 \cap x_2}{x_1 \cup x_2}$ $\rightarrow$ number in both / $\rightarrow$ number in either
  - Distance after dimensionality reduction (later in course).
  - Metric learning (*learn* the best distance function).

Example of KNN

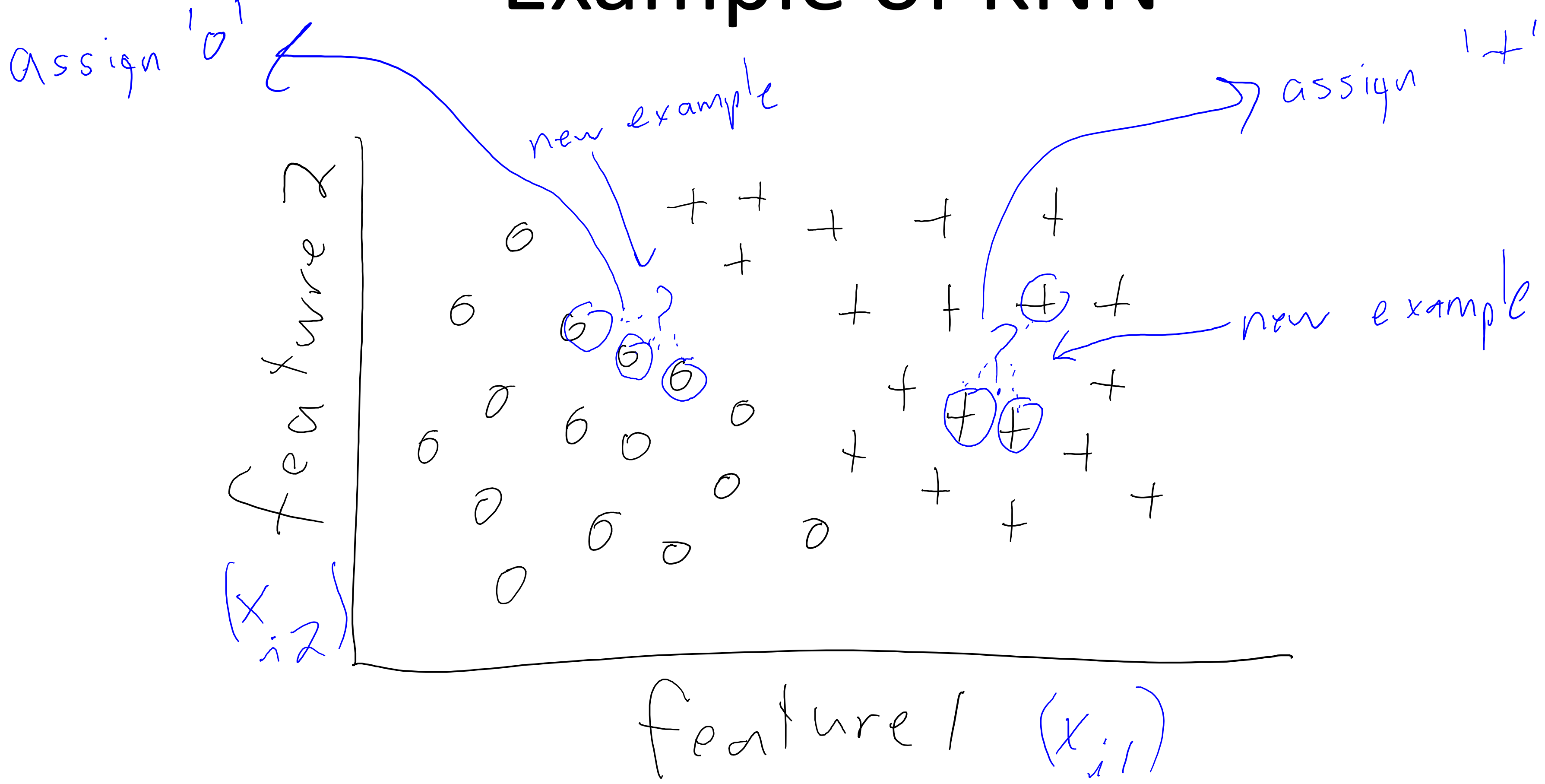
Example of KNN



Example of KNN



Example of KNN



Consistency of KNN

- With a small dataset, KNN model will be very simple.
- With more data, model gets more complicated:
  - Starts to be able to detect subtle differences between examples.
- With a fixed 'k', it has appealing consistency properties:
  - With binary labels and under mild assumptions on distribution:
    - as 'n' goes to infinity, KNN test error is less than twice minimum achievable error.
- Stone's Theorem:
  - If 'k' also goes to infinity and k/n goes to zero:
    - KNN is 'universally consistent': it has the minimum achievable error.
    - Stone's result was the first time any algorithm was shown to have this property.
- Does Stone's Theorem violate the no free lunch theorem?
  - No, Stone's theorem says nothing about performance with finite training set.

Summary

1. Naïve Bayes makes conditional independence assumptions to make estimation practical.
2. Decision theory allows us to consider costs of predictions.
3. Non-parametric models grow the number of parameters with the data set size.
4. K-Nearest Neighbours is a simple non-parametric classifier, with appealing theoretical properties.

- Next Time:
  - Simple tricks to make classifiers work much better.