Unstructured Data Analysis

Lecture 8: Wrap-up topic modeling, intro to predictive data analytics

George Chen
HW1 Questionnaire Results 1

How difficult did you find homework 1?
23 responses

- 1: 0 (0%)
- 2: 5 (21.7%)
- 3: 16 (69.6%)
- 4: 2 (8.7%)
- 5: 0 (0%)
How many hours did you take (roughly) to complete homework 1?

Overall: very good!
HW1 Questionnaire Results III

How do you find the lecture pace?
23 responses

Overall: pacing seems good!
HW1 Questionnaire Results IV

• Some of you want to see more math
  • Check out the supplemental materials!

• I got requests for either a textbook or more supplemental materials:
  • I’m unaware of any textbook that matches the course material right now =( 
  • I hope to revamp supplemental materials in a future version of 94-775 (main problem is finding material at the right technical level)

• Some of you want to see more real-world examples
  • This would be nice! For now, I hope the final projects will get your minds jogging!
Quiz 1 Results

Mean: 75.7, std dev: 13.0, median: 78, max achieved: 100
Quiz Regrade Requests

• How regrades work:

1. Study solutions (already posted in Canvas under “Files”) very carefully

2. If you think there’s a mistake, send me an email and be very specific about what was incorrectly graded and how many points are at stake

3. We will regrade your whole quiz (the version that you submitted to Canvas on the quiz day)

4. Your score can go up, go down, or stay the same, and the regraded result is final

• Due this Friday 11:59pm
HW 1 Regrade Requests

• Same procedure as quiz regrade request

• Also due this Friday 11:59pm
Disclaimer: unfortunately “k” means many things
How to Choose Number of Topics $k$?

Something like CH index is also possible:

For a specific topic, look at the $m$ most probable words (“top words”)

**Coherence (within cluster/topic variability):**

$$\sum_{\text{top words } v, w \text{ that are not the same}} \log \frac{\# \text{ documents that contain both } v \text{ and } w}{\# \text{ documents that contain } w} \log \text{ of } P(\text{see word } v \mid \text{see word } w) + 0.1$$

**Inter-topic similarity (between cluster/topic variability):**

Count # top words that do not appear in any of the other topics’ $m$ top words (number of “unique words”)

Can average each of these across the topics

avoid numerical issues
LDA

Demo
Topic Modeling: Last Remarks

- There are actually *many* topic models, not just LDA
  - Hierarchical Dirichlet process, correlated topic models, anchor word topic models, …

- Dynamic topic models: tracks how topics change *over time*
  - Example: for text over time, figure out how topics change
  - Example: for recommendation system, figure out how user tastes change over time
What if we have labels?
Example: MNIST handwritten digits have known labels
If the labels are known...
If the labels are known…
And we assume data generated by GMM…

What are the model parameters?
Flashback: Learning a GMM

Don’t need this top part if we know the labels!

Step 0: Pick $k$

Step 1: Pick guesses for cluster probabilities, means, and covariances (often done using $k$-means)

Repeat until convergence:

Step 2: Compute probability of each point belonging to each of the $k$ clusters

Step 3: Update cluster probabilities, means, and covariances carefully accounting for probabilities of each point belonging to each of the clusters

We don’t need to repeat until convergence
If the labels are known...

And we assume data generated by GMM...

What are the model parameters?

\[ k = \# \text{ of colors} \]

We can directly estimate cluster means, covariances
What should the label of this new point be?

Whichever cluster has higher probability!
We just created a **classifier**
(a procedure that given a new data point tells us what “class” it belongs to)

This classifier we’ve created assumes a **generative model**

What should the label of this new point be?

**Whichever cluster has higher probability!**
You’ve seen a prediction model that is partly a generative model

Linear regression!
Model parameters: slope $m$, intercept $b$

Feature vector (1D in this case)
For specific value of $x$, assume $y$ drawn from Gaussian with mean $mx+b$, standard dev $\sigma$.
Predictive Data Analysis

Training data

\((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\)

Goal: Given new feature vector \(x\), predict label \(y\)

- \(y\) is discrete (such as colors red and blue)
  - prediction method is called a **classifier**
- \(y\) is continuous (such as a real number)
  - prediction method is called a **regressor**

A giant zoo of methods

- Generative models (like what we just described)
- Discriminative methods (just care about learning prediction rule *without* assuming generative model)
Example of a Discriminative Method: $k$-NN Classification
Example: \textit{k-NN Classification}

What should the label of this new point be?
Example: $k$-NN Classification

What should the label of this new point be?

1-NN classifier prediction
Example: *k*-NN Classification

What should the label of this new point be?

Randomly break tie

2-NN classifier prediction
Example: $k$-NN Classification

What should the label of this new point be?

We just saw: $k = 1$, $k = 2$, $k = 3$

What happens if $k = n$?
How do we choose $k$?

What I’ll describe next can be used to select hyperparameter(s) for any prediction method.

First: How do we assess how good a prediction method is?
Hyperparameters vs. Parameters

• We fit a model’s parameters to training data (terminology: we “learn” the parameters)

• We pick values of hyperparameters and they do not get fit to training data

• Example: Gaussian mixture model
  • Hyperparameter: number of clusters $k$
  • Parameters: cluster probabilities, means, covariances

• Example: $k$-NN classification
  • Hyperparameter: number of nearest neighbors $k$
  • Parameters: N/A
Training data

Want to classify these points correctly

Example: Each data point is an email and we know whether it is spam/ham

Example: future emails to classify as spam/ham
Train method on data in gray

Predict on data in orange

Compute prediction error

Simple data splitting (commonly called train/test split)

In this example: we did a 80%-20% split
Train method on data in gray

Predict on data in orange

Compute prediction error

0% 50%
Train method on data in gray

Predict on data in orange

Compute prediction error

50% 0% 50%
Train method on data in gray

Predict on data in orange

Compute prediction error

0% 50% 0% 50%
Train method on data in gray

Predict on data in orange

Compute prediction error

0% 0% 50% 0% 50%

Average error: \((0+0+50+0+50)/5 = 20\%\)
1. Shuffle data and put them into “folds” (5 folds in this example)

2. For each fold (which consists of its own train/validation sets):
   (a) Train on fold’s training data, test on fold’s validation data
   (b) Compute prediction error

3. Compute average prediction error across the folds
1. Shuffle data and put them into “folds” ($k=5$ folds in this example)

2. For each fold (which consists of its own train/validation sets):
   (a) Train on fold’s training data, test on fold’s validation data
   (b) Compute prediction error

3. Compute average prediction error across the folds

not the same $k$ as in $k$-means or $k$-NN classification
1. Shuffle data and put them into “folds” ($k=5$ folds in this example)

2. For each fold (which consists of its own train/validation sets):
   (a) Train on fold’s training data, test on fold’s validation data
   (b) Compute some sort of prediction score

3. Compute average prediction score across the folds “cross validation score”
Choosing $k$ in $k$-NN Classification

Note: $k$-NN classifier has a single hyperparameter $k$

For each $k = 1, 2, 3, \ldots$, the maximum $k$ you are willing to try:

- Compute 5-fold cross validation score using $k$-NN classifier as prediction method

Use whichever $k$ has the best cross validation score
Automatic Hyperparameter Selection

Suppose the prediction algorithm you’re using has hyperparameters $\theta$

For each hyperparameter setting $\theta$ you are willing to try:

- Compute 5-fold cross validation score using your algorithm with hyperparameters $\theta$

Use whichever $\theta$ has the best cross validation score

Why 5?

People have found using 10 folds or 5 folds to work well in practice but it’s just empirical — there’s no deep reason
**Important:** the errors from simple data splitting and cross-validation are estimates of the true error on test data!

Example: earlier, we got a cross validation score of 20% error

*This is a guess for the error we will get on test data*

This guess is **not always accurate**!

Example: Each data point is an email and we know whether it is spam/ham

Example: future emails to classify as spam/ham
Cross-Validation Remarks

- $k$-fold cross-validation is a randomized procedure
  - Re-running CV results in different cross-validation scores!
- Suppose there are $n$ training data points and $k$ folds
  - If we are trying 10 different hyperparameter settings, how many times do we do model fitting? $10k$
    - If this number is similar in size to $n$, CV can overfit!
  - How many training data are used in each model fit during cross-validation? $\left\lfloor (k-1)/k \right\rfloor n$
    - Smaller # folds typically means faster training
- If $k = n$, would re-running cross-validation result in different cross-validation scores? What about $k = 2$?
  - For deterministic training procedure: same CV result for $k = n$ (since shuffling doesn’t matter), different for $k = 2$