Unstructured Data Analysis

Lecture 10: Intro to predictive data analytics

George Chen
Part I: Exploratory data analysis

*Identify structure present in “unstructured” data*

- Frequency and co-occurrence analysis
- Visualizing high-dimensional data/dimensionality reduction
- Clustering
- Topic modeling

Part II: Predictive data analysis

*Make predictions using known structure in data*

- Classical classification methods
- Neural nets and deep learning for analyzing images and text
What if we have labels?
Disclaimer: unfortunately “k” means many things
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)

Decision boundary

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)

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

Note: Standard linear regression has no generative procedure for generating values of $x$ though!
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?

2-NN classifier prediction

Randomly break tie
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
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 (“proper” training data)  Predict on data in orange (“validation” data)  

Compute prediction error  

Data splitting/“train test split”  
In this example: we did a 80%-20% split
Train method on data in gray ("proper" training data)  Predict on data in orange ("validation" data)

Compute prediction error

50%
Train method on data in gray
("proper" training data)  
Predict on data in orange
("validation" data)

Compute prediction error

0%  
50%
Train method on data in gray ("proper" training data)  
Predict on data in orange ("validation" data)  
Compute prediction error  

50%  0%  50%
Train method on data in gray
("proper" training data)

Predict on data in orange
("validation" data)

Compute prediction error

0% 50% 0% 50%
Train method on data in gray ("proper" training data)  Predict on data in orange ("validation" data)

Compute prediction error

0% 0% 50% 0% 50%

Average error: \( \frac{0+0+50+0+50}{5} = 20\% \)
1. Shuffle data and split them into 5 (roughly) equal size groups
2. For each group $i = 0,1,2,3,4$:
   (a) Treat group $i$ as validation set, the rest as proper training set
   (b) Train on proper training set
   (c) Evaluate prediction error on validation set
3. Compute average prediction error across the folds
1. Shuffle data and split them into 5 (roughly) equal size groups; 
2. For each group $i = 0, 1, 2, 3, 4$:
   (a) Treat group $i$ as validation set, the rest as proper training set
   (b) Train on proper training set
   (c) Evaluate prediction error on validation set
3. Compute average prediction error across the folds called the cross-validation score

not the same $k$ as in $k$-means or $k$-NN classification

$k$-fold Cross Validation
Choosing $k$ in $k$-NN Classification

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
Want to classify these points correctly.

**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? $[(k-1)/k]n$
    - 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 $k = 2$
Different Ways to Measure Accuracy

Simplest way:

- **Raw error rate:** fraction of predicted labels that are wrong (this was in our cross validation example earlier)

In “binary” classification (there are 2 labels such as spam/ham) when 1 label is considered “positive” and the other “negative”: 
Different Ways to Measure Accuracy

Simplest way:

- **Raw error rate:** fraction of predicted labels that are wrong (this was in our cross validation example earlier)

In “binary” classification (there are 2 labels such as spam/ham) when 1 label is considered “positive” and the other “negative”: 
Different Ways to Measure Accuracy

Simplest way:

- **Raw error rate**: fraction of predicted labels that are wrong (this was in our cross validation example earlier)

In “binary” classification (there are 2 labels such as spam/ham) when 1 label is considered “positive” and the other “negative”:
Different Ways to Measure Accuracy

Simplest way:

- **Raw error rate**: fraction of predicted labels that are wrong (this was in our cross validation example earlier)

In “binary” classification (there are 2 labels such as spam/ham) when 1 label is considered “positive” and the other “negative”:

Outlined in dotted black: predicted label + (all other points predicted to be −)

<table>
<thead>
<tr>
<th>True label: −</th>
<th>True label: +</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Blue squares" /></td>
<td><img src="image2.png" alt="Red square" /></td>
</tr>
</tbody>
</table>

Recall/True Positive Rate: fraction of dotted line in true label + = 2/3

Precision: fraction of + in dotted line
- **Raw error rate**: fraction of predicted labels that are wrong (this was in our cross validation example earlier)

In “binary” classification (there are 2 labels such as spam/ham) when 1 label is considered “positive” and the other “negative”:

Outlined in dotted black: predicted label +

(all other points predicted to be −)

**False Positive Rate**: fraction of dotted line in true label −

= 3/7

**Recall/True Positive Rate**: fraction of dotted line in true label +

= 2/3

**Precision**: fraction of + in dotted line

= 2/5

**F1 score**: \[
\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
\]

= 1/2
Prediction and Model Validation

Demo