Feature Engineering

Digging into Data: Jordan Boyd-Graber

University of Maryland

March 4, 2013

COLLEGE OF INFORMATION STUDIES

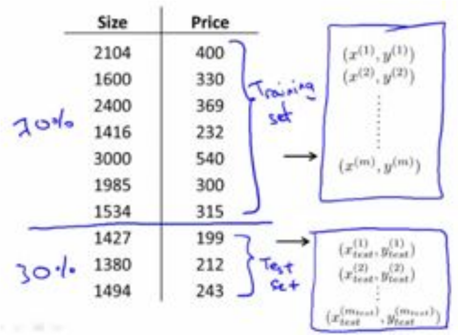
Roadmap

- How to split your dataset
- TV Tropes Dataset
- Feature engineering
- Demo of classification in Rattle

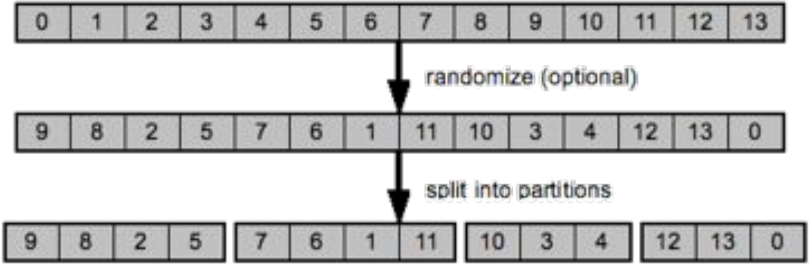
Outline

1. Preparing Data for Classification
2. Evaluating Classification
3. TV Tropes
4. Extracting Features
5. Trying Out Classifiers in Rattle

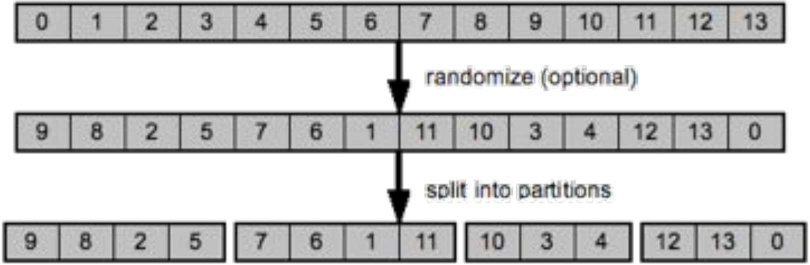
Test Dataset



Partitioning the Data



Partitioning the Data



- Train: Learn a model
- Validation: Evaluate different models
- Test: See how well your model does (only do this once)

Overfitting

Consider error of hypothesis h over

- training data: $error_{train}(h)$
- entire distribution $\mathscr{D}$ of data: $error_{\mathscr{D}}(h)$

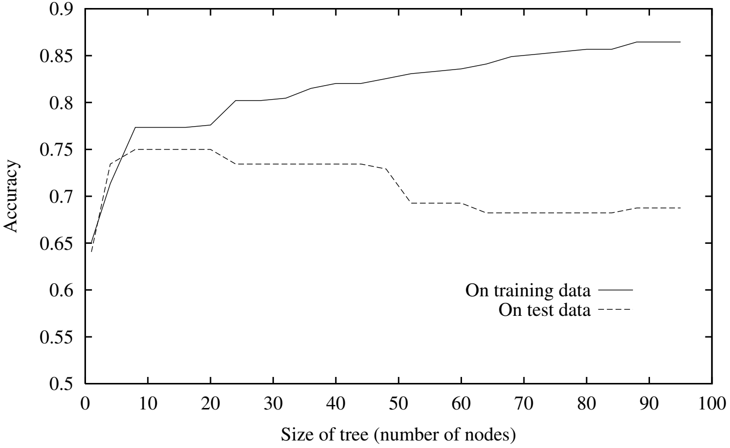
Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

$$error_{train}(h) < error_{train}(h')$$

and

$$error_{\mathscr{D}}(h) > error_{\mathscr{D}}(h')$$

Overfitting in Decision Tree Learning



Avoiding Overfitting

How can we avoid overfitting?

- stop growing when data split not statistically significant
- grow full tree, then post-prune

How to select "best" tree:

- Measure performance over training data to find many models
- Measure performance over separate validation data set to choose one that doesn't overfit

Why validate?

- Often, what you try doesn't work the first time around
  - Process the data somehow
  - Add more features
  - Try different models
- After a while, you get better numbers on your test dataset
- Rattle does this automatically

Outline

Confusion Matrix

	Spam (Predicted)	Non-Spam (Predicted)	Accuracy
Spam (Actual)	27	6	81.81
Non-Spam (Actual)	10	57	85.07
Overall Accuracy			83.44

	p' (Predicted)	n' (Predicted)
p (Actual)	True Positive	False Negative
n (Actual)	False Positive	True Negative

When accuracy lies

	Spam (Predicted)	Non-Spam (Predicted)	Accuracy
Spam (Actual)	0	10	0.0
Non-Spam (Actual)	0	990	100.0
Overall Accuracy			99

When accuracy lies

	Spam (Predicted)	Non-Spam (Predicted)	Accuracy
Spam (Actual)	0	10	0.0
Non-Spam (Actual)	0	990	100.0
Overall Accuracy			99

Moral: If you care about X, make sure your data have it!

Outline

TV Tropes

- Social media site
- Catalog of “tropes”
- Functionally like Wikipedia, but ...
    - Less formal
    - No notability requirement
    - Focused on popular culture

Absent-Minded Professor

- “Doc” Emmett Brown from *Back to the Future*.
- The drunk mathematician in *Strangers on a Train* becomes a plot point, because of his forgetfulness, Guy is suspected of a murder he didn’t commit.
- *The Muppet Show*: Dr. Bunsen Honeydew.

Spoilers

- What makes neat is that the dataset is annotated by users for **spoilers.**
- A spoiler: “A published piece of information that divulges a surprise, such as a plot twist in a movie.”

Spoiler

- Han Solo arriving just in time to save Luke from Vader and buy Luke the vital seconds needed to send the proton torpedos into the Death Star’s thermal exhaust port.
- Leia, after finding out that despite her (feigned) cooperation, Tarkin intends to destroy Alderaan anyway.
- Luke rushes to the farm, only to find it already raided and his relatives dead harkens to an equally distressing scene in The Searchers.

Not a spoiler

- Diving into the garbage chute gets them out of the firefight, but the droids have to save them from the compacter.
- They do some pretty evil things with that Death Star, but we never hear much of how they affect the rest of the Galaxy. A deleted scene between Luke and Biggs explores this somewhat.
- Luke enters Leia’s cell in a Stormtrooper uniform, and she calmly starts some banter.

The dataset

- Downloaded the pages associated with a **show.** Took complete sentences from the text and split them into ones with spoilers and those without
- Created a balanced dataset (50% spoilers, 50% not)
- Split into training, development, and test **shows**

The dataset

- Downloaded the pages associated with a **show.** Took complete sentences from the text and split them into ones with spoilers and those without
- Created a balanced dataset (50% spoilers, 50% not)
- Split into training, development, and test **shows**
    - Why is this important?

The dataset

- Downloaded the pages associated with a **show.** Took complete sentences from the text and split them into ones with spoilers and those without
- Created a balanced dataset (50% spoilers, 50% not)
- Split into training, development, and test **shows**
  - Why is this important?
- I'll show results using SVM; similar results apply to other classifiers

Outline

Step 1: The obvious

- Take every sentence, and split on on-characters.
- Input: “These aren’t the droids you’re looking for.”

Step 1: The obvious

- Take every sentence, and split on on-characters.
- Input: “These aren’t the droids you’re looking for.”

Features

These:1 aren:1 t:1 the:1 droids:1 you:1 re:1 looking:1 for:1

	False	True
False	56	34
True	583	605

Accuracy: 0.517

Step 1: The obvious

- Take every sentence, and split on on-characters.
- Input: “These aren’t the droids you’re looking for.”

Features

These:1 aren:1 t:1 the:1 droids:1 you:1 re:1 looking:1 for:1

What’s wrong with this?

	False	True
False	56	34
True	583	605

Accuracy: 0.517

Step 2: Normalization

- Normalize the words
  - Lowercase everything
  - Stem the words (not always a good idea!)
- Input: “These aren’t the droids you’re looking for.”

Step 2: Normalization

- Normalize the words
  - Lowercase everything
  - Stem the words (not always a good idea!)
- Input: “These aren’t the droids you’re looking for.”

Features

these:1 are:1 t:1 the:1 droid:1
you:1 re:1 look:1 for:1

	False	True
False	52	27
True	587	612

Accuracy: 0.520

Step 3: Remove Usless Features

- Use a “stoplist”
- Remove features that appear in > 10% of observations (and aren’t correlated with label)
- Input: “These aren’t the droids you’re looking for.”

Step 3: Remove Usless Features

- Use a “stoplist”
- Remove features that appear in > 10% of observations (and aren’t correlated with label)
- Input: “These aren’t the droids you’re looking for.”

Features

droid:1 look:1

	False	True
False	59	20
True	578	621

Accuracy: 0.532

Step 4: Add Useful Features

- Use bigrams (“these_are”) instead of unigrams (“these”, “are”)
- Creates a lot of features!
- Input: “These aren’t the droids you’re looking for.”

Step 4: Add Useful Features

- Use bigrams (“these_are”) instead of unigrams (“these”, “are”)
- Creates a lot of features!
- Input: “These aren’t the droids you’re looking for.”

Features

these_are:1 aren_t:1 t_the:1 the_droids:1 you_re:1 re_looking:1 looking_for:1

	False	True
False	203	104
True	436	535

Accuracy: 0.578

Step 5: Prune (Again)

- Not all bigrams appear often
- SVM has to search a long time and might not get to the right answer
- Helps to prune features
- Input: “These aren’t the droids you’re looking for.”

Step 5: Prune (Again)

- Not all bigrams appear often
- SVM has to search a long time and might not get to the right answer
- Helps to prune features
- Input: “These aren’t the droids you’re looking for.”

Features

these_are:1 the_droids:1
re_looking:1 looking_for:1

	False	True
False	410	276
True	229	363

Accuracy: 0.605

How do you find new features?

- Make predictions on the development set.
- Look at contingency table; where are the errors?
- What do you miss?

How do you find new features?

- Make predictions on the development set.
- Look at contingency table; where are the errors?
- What do you miss? **Error analysis!**
- What feature would the classifier need to get this right?
- What features are confusing the classifier?
  - If it never appears in the development set, it isn't useful
  - If it doesn't appear often, it isn't useful

How do you know something is a good feature?

- Make a contingency table for that feature (should give you good information gain)
- Throw it into your classifier (accuracy should improve)

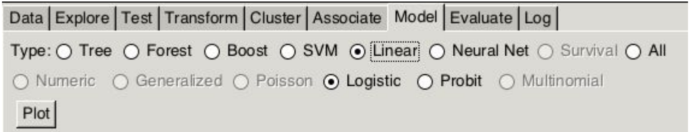
Homework 2

- I've given you TV Tropes data
- And development data
- And test data (no labels)
- Only have 15 features (should get you around 56%)
    - For these features, it doesn't matter (much) which classifier you use
- Your job: add additional features and see how they do

Outline

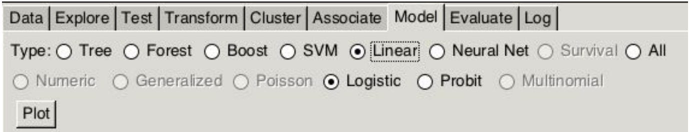
Selecting a model

- Go to “model” tab and select one of the models
- Make sure the model makes sense
- For logistic regression, select “linear” and “logistic”

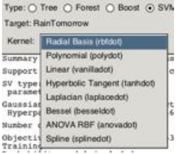


Selecting a model

- Go to "model" tab and select one of the models
- Make sure the model makes sense
- For logistic regression, select "linear" and "logistic"

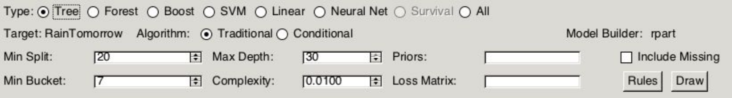


- For SVM, you also need to select a kernel (try linear first, then "Gaussian" which will be much slower)



- Output varies by model
  - SVM is least informative (hard to summarize)
  - Note you can click **draw** to see decision trees

Decision Trees Have Many Options . . .



- **Prior**: The prior observation probabilities (in case your training data are skewed)
- **Min Split**: How many observations can be in an expanded leaf (pre-test)
- **Min Bucket**: How many observations can be in any resulting leaf (post-test)
- **Max Depth**: How many levels the tree has
- **Complexity**: How many “if” statements the tree has

Defaults are reasonable; tweak if you are having complexity issues.

How'd we do?

- Fit the model by clicking on the "execute" button

- Click on the evaluate tab, have your boxes checked for the models you want to compare
- Select specific datasets (e.g. external csv file)
- For the weather dataset, SVM does best (.14)
- To get explicit predictions, click the score button
- We'll learn about the other metrics next week!

RTextTools

```
library(RTextTools)

train.df <- read.csv("train/train.csv")
train.df$sentence <- as.character(train.df$sentence)

dev.df <- read.csv("dev/dev.csv")
dev.df$sentence <- as.character(dev.df$sentence)

train.df <- train.df[1:1000,]
dev.df <- dev.df[1:100,]

data <- rbind(train.df, dev.df)
dev_size <- dim(dev.df)[1]
total_size <- dim(data)[1]

matrix <- create_matrix(cbind(data$sentence, data$trope),
                language="english", removeNumbers=TRUE, stemWords=FALSE,
                weighting=weightTfIdf)

container <- create_container(matrix, data$spoiler, trainSize=1:dev_size,
                       testSize=(1+dev_size):total_size, virgin=FALSE)

models <- train_models(container, algorithms=c("MAXENT","SVM"))
results <- classify_models(container, models)
```