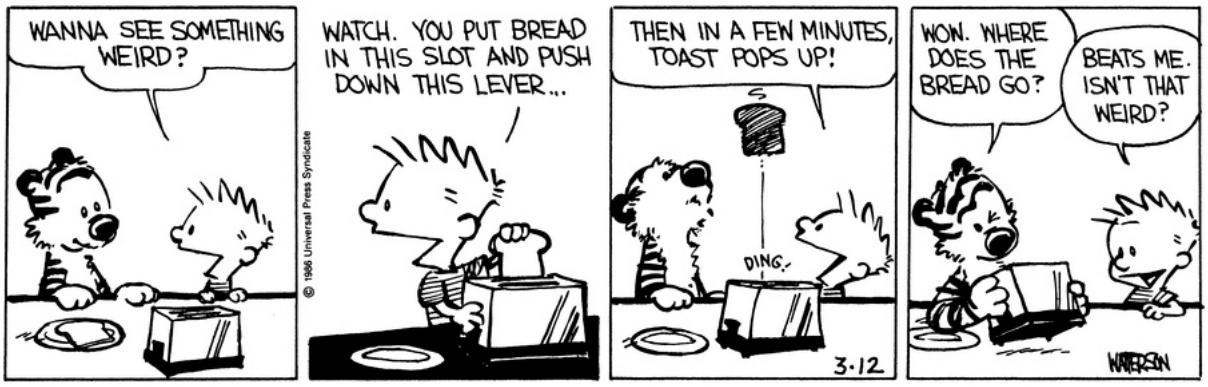
CS159 Machine Learning Lab



https://www.gocomics.com/calvinandhobbes/1986/03/12

In this class/lab today, we're going to be playing with some of the machine learning techniques that we've been talking about in class, in particular, we're going to compare Naive Bayes (NB) versus multinomial logistic regression (MLR or maximum entropy classifier) for name gender identification. Although we didn't talk about explicitly about the MLR model in class, it's one of the models that tends to perform very well across a broad range of tasks.

Both of these models allow us to classify multiple examples and both models are also probabilistic classifiers. However, there are some difference. NB is a *generative* model, modeling the joint distribution of $p(label, features)$ and NB also has the naive Bayes assumption that the features are independent given the class. MLR on the other hand is a *discriminative* model, modeling the conditional distribution $p(label|features)$ directly and does not have the strong assumptions of NB.

Name gender identification

Today, we're going to try and classify first names as either male or female using machine learning. In general, gender is not binary, but our dataset is limited to these two annotations. While names are fluid and can be used for any gender, in practice, certain names tend to be used by certain genders. Our goal is to try and use machine learning techniques to identify these general trends and not to suggest that a name must be used/associated with a particular gender. Our dataset does capture some of the flexibility of name usage and for some names does have then annotated as both male and female.

Just to give you a feeling for what the task is and how well people can do it, try and label the following ten names as most commonly used for either male or female (**don't peek at the next page until you've got your answers!**):

Nettie
Darci
Zalman
Stefa
Caprice
Nikos
Elisa
Winn
Marlyn
Harman

Nettie	female
Darci	female
Zalman	male
Stefa	female
Caprice	female
Nikos	male
Elisa	female
Winn	male
Marlyn	female
Harman	male

Do you agree with the annotations? For those you didn't have a previous association with, how did you know/guess that they were male or female?

Getting Started

To get started:

- Open `Terminal` and create a directory in your home directory called `ml_lab`.

  ```
  mkdir ml_lab
  ```

- Copy the contents for today's lab into that directory

  ```
  cp -r /common/cs/cs159/assignments/ml_lab/* .
  ```

  If you're working on your own laptop, use `scp`

  ```
  scp -r <username>@little.cs.pomona.edu:/common/cs/cs159/assignments/ml_lab/* .
  ```

  where `<username>` is your Pomona CS username.

Data

In the `data` directory you'll see three files:

- `names.train` a file containing approximately 7,000 names for training
- `names.dev` a development set of 500 names
- `names.test` a test set of 500 names (**Do NOT look at this or use this until I tell you to!**)

Use `grep` to separate the male names and the female names. Spend a few minutes looking at the names. Are there certain characteristics that female/male names have? How are the names different? This should motivate features you might use later on.

Code

In the `code` directory I have included some code to get you started classifying these examples. You may develop your code either in `Eclipse` or via the command-line. To get started in `Eclipse`:

1. Create a new Java project in `Eclipse`
2. copy the .java files into your source directory and refresh the project
3. Right-click on the project and select "Build Path -> Configure Build Path ..."
4. Select the "Libraries" tab and then click the "Add External JARs..." button. Browse to `ml_lab/code/` and select `stanford-classifier.jar`.

To get started from the command-line:

1. Change directories into the `code` directory.
2. To compile, type:

   ```
   javac -classpath stanford-classifier.jar:.  *.java
   ```

3. and to run:

   ```
   java -classpath stanford-classifier.jar:.  NameClassification
   ```

The current models

The starter code trains a NB model and a MLR model on the training data using 26 features based on the occurrence of each of the 26 letters in the name. The code then classifies the development set and prints out the classification accuracy and the log-prob for both of the models. You'll need to change the `DATA_DIR` constant to point at your data directory.

I've left in the call to `.dump()` for the MLR classifier to see what the weightings for the different features are. The magnitude indicates how important the weight/feature is and the sign, whether or not it is an indicator or an inhibitor for that class. Which feature(s) are most indicative of each class? We can also call `.dump()` on the NB model. Are the most indicative features similar?

The best model

For the rest of this lab, I'd like you to try and come up with what you think is your best classifier for both NB and MLR. Do *NOT* look at the test data yet. You should only be playing with the training and development data. The best way to improve your model is to include more features. You can include very specific individual features or you can also include other "groups" of features like I've done with the letter occurrence features.

To add your features, you'll mostly be modifying the `examplesToData` method, though feel free to write other supporting methods, etc.

Be creative and analyze the mistakes that your model is making as well as the parameters learned. If a particular set of features tends to have low weights overall, it may just be confusing the model. The `accuracy` method has a boolean that can be set to `true` to print out the examples that are misclassified. This may also prove useful.

You might also try playing with some of the smoothing parameters for the models, but don't do this until you're pretty comfortable with your features. I've posted the documentation for the classifier code at:

`http://www.cs.pomona.edu/classes/cs159/lectures/ml_lab/documentation/`

When we have about 10-15 minutes left of class, we'll compare the performance of your classifiers on the test set and talk about different features.

Training data size

If you have time spend 5 or 10 minutes exploring the performance of the classifiers for varying amounts of training data. We saw in class that NB tends to perform better for small amounts of data and MLR as we get more training data. Does that hold for us? Would varying the number of features affect your analysis?