

Machine Learning for Text

July 29, 2015

1 Intro to Supervised Learning

- * intro, purpose to ML
- * ML pipeline, setup, feature matrix
- * example of data
- * heuristic rules
- * from heuristics to decision trees
- * from heuristics to linear regression
- * popular packages

2 ML algorithms for supervised learning

2.1 Decision Trees

2.2 Linear and Logistic Regression

2.3 Other Supervised Algorithms

3 ML or IR, evaluation, cross validation

3.1 running LibLinear

Liblinear

LibLinear is a public Machine Learning package for linear classifier for data with millions of instances and features. You could download the package from <http://www.csie.ntu.edu.tw/~cjlin/liblinear/>.

Once you have the unzipped folder of LibLinear, you need first compile the files. To do this, just run "make" in your terminal.

```
[bingyu@fiji11 ~]$ cd liblinear
[bingyu@fiji11 liblinear]$ ls
blas          heart_scale  linear.def   Makefile     matlab       python       train.c      tron.h
COPYRIGHT    linear.cpp   linear.h     Makefile.win predict.c     README       tron.cpp     windows
[bingyu@fiji11 liblinear]$ make
g++ -Wall -Wconversion -O3 -fPIC -c -o tron.o tron.cpp
g++ -Wall -Wconversion -O3 -fPIC -c -o linear.o linear.cpp
make -C blas OPTFLAGS='-Wall -Wconversion -O3 -fPIC' CC='cc';
make[1]: Entering directory `/home/bingyu/liblinear/blas'
cc -Wall -Wconversion -O3 -fPIC -c dnm2.c
cc -Wall -Wconversion -O3 -fPIC -c daxpy.c
cc -Wall -Wconversion -O3 -fPIC -c ddot.c
cc -Wall -Wconversion -O3 -fPIC -c dscal.c
ar rcv blas.a dnm2.o daxpy.o ddot.o dscal.o
a - dnm2.o
a - daxpy.o
a - ddot.o
a - dscal.o
ranlib blas.a
make[1]: Leaving directory `/home/bingyu/liblinear/blas'
g++ -Wall -Wconversion -O3 -fPIC -o train train.c tron.o linear.o blas/blas.a
g++ -Wall -Wconversion -O3 -fPIC -o predict predict.c tron.o linear.o blas/blas.a
[bingyu@fiji11 liblinear]$
```

Figure 1: Compile the LibLinear Package

After compiling, you could do training and prediction process. Suppose your input feature matrix named “train.txt” and “test.txt”. To do the simple training and prediction:

training

Run “./train train.txt linear.model” in your terminal.

“train.txt” is your input training feature matrix file. “linear.model” is the output model name, you could name it.

prediction

Run “./predict test.txt linear.model linear.predict”.

“test.txt” is your input testing file. “linear.model”, this is the model you got from training process as an input here. “linear.predict” is the output prediction results for testing data. You could name it.

```
[bingyu@fiji11 liblinear]$ ./train train.txt linear.model
.....
optimization finished, #iter = 1000

WARNING: reaching max number of iterations
Using -s 2 may be faster (also see FAQ)

Objective value = -1.056825
nSV = 1171
[bingyu@fiji11 liblinear]$ ./predict test.txt linear.model linear.predict
Accuracy = 99.8674% (15064/15084)
[bingyu@fiji11 liblinear]$
```

Figure 2: Training and Prediction by LibLinear Example

3.2 ML evaluation

To evaluate your Machine Learning algorithms, first you need select a training set and testing set. There is no overlapping between these two groups, which means you could not include any testing samples in your training, or any training samples in your testing. Random selecting from data into training and testing could be a simple way.

After getting training and testing, the Machine Learning algorithms will be trained on training set and will be evaluated on the testing set. Remember that the model training process is not allowed to access any testing set.

Once you have the trained model, you could do prediction on testing set using the trained model. To evaluate the predictions on the testing set, based on different Machine Learning tasks, there are different performance measures. For classification, we could choose accuracy (total number of correct predictions on testing divided by the total number of testing samples.) Furthermore, to dig more information from accuracy, you could refer confusion matrix(https://en.wikipedia.org/wiki/Confusion_matrix) information. For example, in spam classification, people may care more about the false positives. For regression problem, you may consider the Root Mean Squared Error as the measure(https://en.wikipedia.org/wiki/Mean_squared_error).

3.3 Cross Validation

A more sophisticated way than evaluate ML only using one training and testing set is trying to randomly split the entire dataset into a few folders evenly, let's say 5 folders. Then you could do following evaluations:

- Train on folder 1,2,3,4; Test on 5
- Train on folder 2,3,4,5; Test on 1
- Train on folder 3,4,5,1; Test on 2
- Train on folder 4,5,1,2; Test on 3
- Train on folder 5,1,2,3; Test on 4

Finally you could take an average of your five times performance measures on testing set as your final evaluation. We call this method as Cross Validation.

The number of folders is chosen based on the size of your dataset. We usually choose 5 or 10.

4 ML for text, IR data

4.1 Document Features

We could regard documents or text as sequences of words so far, but documents have much richer structure and information. Let us see some other extra features we can use as clues of relevance.

Structural Features

Some of these features are structural: the document's organization gives clues about the topic:

- The title, headings, and menu give fine-grained topic and subtopic information.
- Links and their anchor text provide clues about the relevance of other pages related to this one.

The image shows a screenshot of a Wikipedia article for Susan Dumais. The page layout includes a sidebar on the left with navigation links, a main content area, and a search bar at the top right. Several elements are highlighted with blue dashed boxes and labels:

- Title:** The article title "Susan Dumais" is highlighted.
- Bold Text:** The word "Bold Text" is highlighted in a blue box, pointing to the bolded text "Susan Dumais" in the first sentence of the article.
- Links:** The word "Links" is highlighted in a blue box, pointing to the word "information retrieval" in the text.

The article text includes a biographical paragraph, a photo of Susan Dumais, and a paragraph discussing her research on the "vocabulary problem" in information retrieval, leading to the invention of Latent Semantic Indexing.

http://en.wikipedia.org/wiki/Susan_Dumais

Figure 3: Structural Features Example

Topical Features

Other features are topical: the document's text may contain special words and phrases that pertain to a certain topic.

- Named entities (people, companies, places, events ...) are strong topical clues.
- Topic modeling discovers the vocabulary that tends to be used when speaking of a certain topic.

The image shows a screenshot of the Wikipedia article for Susan Dumais. The page layout includes the Wikipedia logo and navigation tabs at the top. The main text describes her as a Distinguished Scientist at Microsoft and an Affiliate Professor at the University of Washington Information School. A photo of her in 2009 is included. Two blue boxes with dashed borders highlight specific terms: 'Entities' and 'Topical Terms'.

http://en.wikipedia.org/wiki/Susan_Dumais


Figure 4: Topical Features Example

Features from ML

Tools from Machine Learning can be used to generate additional features for a page.

- Document classifiers are used to identify news articles, blogs, reviews, and other types of specialized pages.
- Document clustering can find very similar pages, which is useful for providing diverse result lists and “more like this” functions (e.g. clustering news articles by story).

Create account Log in



WIKIPEDIA
The Free Encyclopedia

- [Main page](#)
- [Contents](#)
- [Featured content](#)
- [Current events](#)
- [Random article](#)
- [Donate to Wikipedia](#)
- [Wikimedia Shop](#)

Interaction

- [Help](#)
- [About Wikipedia](#)
- [Community portal](#)
- [Recent changes](#)
- [Contact page](#)

Tools

- [What links here](#)
- [Related changes](#)
- [Upload file](#)
- [Special pages](#)
- [Permanent link](#)
- [Page information](#)
- [Wikidata item](#)
- [Cite this page](#)


[Print/export](#)

Document Class: Biographical

Susan Dumais

From Wikipedia, the free encyclopedia

Susan Dumais is a Distinguished Scientist at Microsoft and manager of the Context, Learning, and User Experience for Search (CLUES) Group of [Microsoft Research](#). She is also an Affiliate Professor at the [University of Washington Information School](#).



Susan Dumais in 2009 in her office at Microsoft Research.

Before joining Microsoft in 1997, Dumais was a researcher at Bellcore (now [Telcordia Technologies](#)), where she and her colleagues conducted research into what is now called the **vocabulary problem** in [information retrieval](#).^[1] Their study demonstrated, through a variety of experiments, that different people use different vocabulary to describe the same thing, and that even choosing the "best" term to describe something is not enough for others to find it. One implication of this work is that because the author of a document may use different vocabulary than someone searching for the document, traditional [information retrieval](#) methods will have limited success.

Dumais and the other Bellcore researchers then began investigating ways to build search systems that avoided the vocabulary problem. The result was their invention of [Latent Semantic Indexing](#).^[2]

http://en.wikipedia.org/wiki/Susan_Dumais

Figure 5: Features from ML Example

4.2 Feature Matrix

When we have collected all the document features we are interested in, we can use standard Machine Learning classifiers to learn how to predict document relevance from document features. This makes scoring functions such as BM25 simply one component of a more complicated relevance predictor. And all documents and queries can be converted into numeric vectors that provide this rich information to learning algorithms. This allows us to leverage the best ML techniques for document ranking. An example is shown as below:

DocID	Body BM25	Title BM25	In-links	...	Rel?
1	1.23	1.2	3		1
2	1.2	1.4	12		1
3	17.3	13.2	2		0
4	10.55	0	207		0

Figure 6: Features Matrix Example

4.3 setup for IR data

- * split TRAIN/TEST across queries. Learning to rank setup
- * sort the output per query
- * compute IR measures

5 ES feature value collection, sparse Feature Matrix

5.1 ES function calls for feature values “span near query”

5.2 enumerate unigrams, bigrams

5.3 Sparse Feature Matrix

- *Cheng’s format on disk, including auxiliary files
- *Sparse Matrix in memory

5.4 skip-grams, slop

5.5 Running Cheng’s Learning Algorithms

6 Data Sampling, Feature Selection, Regularization

- * overfitting
- * feature selection
- * regularization L2
- * regularization L1
- * data sampling

7 Feature Analysis

* top features for regression

* top features for Decision Trees