Sigmoid convert to probability

SGD: stochastic gradient descent

Just running optimization may lead to overfitting (assumptions that does not generalize)

Cause of error:

* Assign weight to all features even little influence ones
* Assign very high weight

Add prior weight

Ignore no evidence: penalize nonzero values (L1 regularization)

Maximum entropy model: logistic regression classifier

Measure regularized log likelihood (by having regularization)

Linear regression classification

Regular expression tokenizer or just spaces to represent each index in the word vector

Inverse document frequency taking log for coherent range

Make sure your machine learning is sensitive to distinctions. How to weigh the feature. In the case of sentiment, certain additive may be more important such as boring/ exciting. Make sure vocabulary is treated accordingly. First, we can do nothing. The learner can see and count the word in the evidence collection. There are exceptions. One has to do with sparsity. You may not be able to get a big enough of data to get a big enough collection about rare words.

Prior knowledge can come in. But make sure the data is sparse or it will obscure before you build the vocabulary.

Ambiguity: a soda is cool may be just describing the temperature. You should Apply on processed text/ tagging information that says more about the document.

Fit transform in sklearn TFIDF set weight for other: use training data, processing , save, apply on dev, part of learning.

LDA is unsupervised.

Classification problem vs clustering problem.

Classifier: Confusion\_matrix

When half and half, can use accuracy and error

When incidents are rare, use precision and recall

Recall: True Positive over all yes things.

The yes answers are relatively rare. It’s often challenging to get high precision and recall but with the policy of “NO!” you can get high accuracy.

set higher precision/ threshold. Saying yes less often but saying yes more accurately.

set higher recall, set lower threshold.

Use ROC\_curve to visualize.

Naïve Bayes classifier

Base line: understanding the task.

Ceiling:

Measuring performance: partly random.

Optimizer’s curse: may get better result out of accident. Hands off test data.

Simulation of monte carol

Logistic regression classifier is better than Naïve Bayes classifier

Test the power of the experiment

The power of experimental result to distinguish between

Do I have data to draw conclusion

100

75% CL1 O Y N N Y swap last 2 col Y N Y Y 50%

0% CL2 O N N Y Y N N N Y 25%

Truth O Y N N N

Feature 1 labeled 1, feature 2 labeled 0.6, 0.4,0.3,0.2

1 0.6 0.4 0.3 0.2

T F F F

It takes three of the rest features to overcome

Brown corpus in NLTK

Word frequency scatter plot : apply loglog to give linear curve

Heavy tail distribution/ long tail (infrequent items)

Log normal distributions, appear to be linear

Word presence/absence vector more meaningful?

Supervised classification problem.

Pointwise mutual information

Auto complete

Meet + me/us/him

Does not necessarily mean meet+ me happen together.

San + Francisco this is from PMI two rare things go together

Bigram: nltk manages token stream , frequency of words, denominator

Wordnet: a database

Word 🡪 synsets set of synonyms

Meronym mero = part

Text Classification

Put it in bins

Sentiment analysis: classifying document as positive or negative

Classifying by topic

Because text classification is important and useful. What you have to add to make a Text classification problem a good project. (Instead of two lines python)

There are a lot of cases. You have to do something more sophiscated. Get text classifier working in good way. In two directions:

1. System architecture: if you take this text classification learning mechanism. Such as improving user interaction in a way. This turns out to be where NL technology can be ???

Mail systems capturing people’s reaction from document and learn them. System building problem new prospective interpret of data. Improve interaction/ ability to support people who ask for info.

Get the data&use it creatively.

1. Features: the features are super easy to get. Count tokens. Characters occur between spaces. There are lots of cases that worthwhile more interesting features giving you opportunity to more interesting language processing.

A challenging language (Not English). Where you need to do word segmentation. Implement this “tagging” problem. Complex morphology like Arabic. There are rules for forming orders.

In language like English, it’s hard to beat the basic features. If you have an idea for a particular new feature, worth trying.

Or anything else that requires tagging or other deeper NLP inference

These are two independent directions to go. No need to combine them.

The key thing is text classification is a starting point. Your project should have something new. Word features/embedding, logistic regression classifier.

Study problems are easy to handle with shallow techniques.

It’s time to start thinking things you are interested in.

Classification/Tagging / Translation(we did not look at translation)

Other techniques for word distribution and similarity of document???

Latent-

Statistical ways to capture the ways that documents differ

Regression model based on their vector representation.

Topic modelling: based on statistical Latent Dirichlet Allocation

Statistical model: compress the collection as much as possible, common factors. By thinking or documents address a mixture of topics. Words picked randomly based on the topic.

A topic is a latent variable: you infer as abstract ???

For each topic, you have a distribution over words.

The way you write a document in LDA model, you go one ???

Hidden dimensions of the topic.

Some documents can share, links some words together.

Compare to methods like Glove or Word2Vec. This topic model is relatively expensive to build. People usually build it on large. You do not have to consider all possible words.

Example lda-demo

Fitting LDA models: infer representation of each document as a probability of each topic. How often it would expect to see different words in each topic given the document so far.

We told it to have 10 different topics.

Probabilistic weighting how important each document is to each topic.

Term-frequency matrix (2000,1000) to learn from

Compressed version (2000,10)

Randomly assign words to topics. Iterative fitting process. Variational MCMC

This is an optimization problem. Objective function is the Log likelihood the model assigns to the training data.

Computing the similarity between documents or giving insights of word organization and collections of documents. What is trending over time. Real structure of the scientific ??? who likes each other.

Cross cutting what words are used.

Look for documents inn different attributes. You may look for dates or particular sources/methods by looking at this model.

The representation for each word in this model is basically a vector of 10 numbers. The number represents the expected counts for this word across different topics. You can compute similarity between different words by a dot product.

Overlap of topics. Also words go together because they have similar syntactic behavior. This statistical similarity may be different.

Previously we are not using order of words at all.

Speech tagging problem. Labeling the use of a particular occurrence of a word based on some aspect of the row???

A Locally ambiguous example: taking these words in tokens and how best to take the sequence into account?

Organize the decisions you give to the machine learner. The learner can use he context to make decisions.

There are interesting things about how you deal with streams of data: take your dataset, convert it to training sets, take tokens in context.

Combining learning and search. Try to find entire sequence at once. Overall consistent labeling.

Diminishing returns.

P(w2|w1)

co-occurrence

P(the|START) high

P(END|.) high

PRON BERB PRT VERB

They refuse to permit

Hidden Markov Model

Underlying model

Transition matrix for each category anytime in sequence what is the probability to get any other category in sequence.

Markov assumption

P(Ti+1|Ti) there is no dependency across time until the next

P(wi|Ti)

Strong independence between words. Knowing the previous words does not help you anything.

There are pairs of words that actually go together.

Sequence tagging a search

Building a model quickly

Counting the tags that follows you can get the next text in sequence.

You do not know what it means you have not seen something. It does not mean it will never happen. Assign some non-zero to things you have not seen.

Infer sensible numbers

In every machine learning problem you have to be designing for new things that you have not seen.

We are observing the words start at the beginning and end.

We want distribution.

Pick the sequence of the tags that maximize the probability of what we have observed.

P(PR|ST)

P(they|PR)

P(VB|PR)

P(refuse|VB)

P(spvpve,trftp)

A particular tag sequence and a particular words

P(tags|words)=P(tags,words)/P(words)

Maximize P(tags|words)= maximize P(tags,words) since P(words) is given.

Get the probability from the model definition.

In principle, you could conceivably solve it by???

Suppose you know you have a PRT at a particular time. We know there is a score associated with this edge.

They refuse to permit.

Intuition is at any point of calculation, you only need to remember the most likely way of how to get to this point. Imagine you have two paths before. They may have different score.

Whatever happens downstream of here is numerical score

What is the overall score of the two paths. Given you want to end up at the PRT, the path before it should also be maximized, discard partial worse path.

Incremental path

Bear Habitat

N 🡪 N

V 🡪

NN link is more likely than the VN. (counted in the corpus)

Given that this habitat is a N, this note is given. The following info will never affect bear. There is no other influence to void previous state. When you are looking at a node, once you know where the path is, it’s enough to take the best path. The best path is inclusive (model have scores for different paths.)

Crystal clear vs glass clear

N 🡪 ADT

P(Crystal|N) What is the probability a N is crystal

P(ADT|N)

P(clear|ADT)

The model will get it wrong because it does not consider the dependency of the words.

P(Glass|N)

P(ADT|N)  
P(clear|ADT)

It’s possible to train this model that you think this P is weights not probability. Just scoring.

The model only tracks the relation between categories, not words.

Simple to train and simple to compute with.

Only local info needed to disambiguate.

Shape of one dynamic programming algorithm: Viterbi

Keep a pointer to the best analysis and keep the score of the best analysis up to that point.

They refuse to permit

🡨Noun NOUN

🡨PRON PRON

🡨VERB VERB

…

Multiplying the terms together.