L101: Machine Learning for Language Processing

Lecture 1

Guy Emerson

About the course

- Introduction to using Machine Learning for Natural Language Processing
- Prerequisites:
  - L90 (or similar) – essential
  - L95 – desirable
- 8 lectures, 8 seminars, 1 essay/project

The focus is on *using* Machine Learning – this is not a pure Machine Learning course, but rather an NLP course where Machine Learning provides a set of tools.

I (Guy Emerson) will give 7 of the lectures.
Ted Briscoe will give 1 lecture, the 8 seminars, and assess the essay or mini-project.

Sources of Information

- Course web pages
  - Handouts include additional notes!
- L90 (and L95) notes
- Textbooks, e.g. Jurafsky & Martin
- Ask questions!

Jurafsky & Martin, draft 3rd edition available online:
`https://web.stanford.edu/~jurafsky/slp3/`

Today’s Lecture

- What is Machine Learning?
- Example: topic classification
- How do we know if it works?

What is Machine Learning?

- Task
- Data
- Model
- Training

We can break down Machine Learning into four parts, which I will go through one at a time.

Tasks

- What do we want to do?
- Abstract from a real-world problem
- Examples:
  - Sentiment analysis
  - Topic classification
  - Machine translation

The first step is to define the task. This includes "non-technical" questions, such as who the intended user is (a researcher? a company? a member of the public?), which can be important in deciding how to formalise the task. This step is often the least discussed, but in real-world applications, it can be the most difficult.

Defining a task usually requires simplifying a real-world problem. Sentiment analysis (for example, deciding if a product review is positive or negative) simplifies the complexity of possible opinions into a binary value. Topic classification simplifies the complexity of possible topics into a small set. Machine translation might at first look like a real-world problem (after all, people translate things!), but it usually involves translating one sentence at a time, without any context, and is usually evaluated using BLEU scores, rather than considering the purpose of the document. Further reading on BLEU: Callison-Burch et al. (2006) `http://aclweb.org/anthology/E06-1032`

Data

- Types of data:
  - Natural (e.g. “raw” text)
  - Pre-processed (e.g. tokenised text)
  - Annotated (e.g. pos-tagged text)

It almost goes without saying that Machine Learning requires machine-readable data.

Pre-processing refers to automated data preparation.

Annotation is something in addition to the text itself, such as part-of-speech tags, parse trees, and so on. Text might be manually annotated (by experts or by crowdworkers) or automatically annotated (by existing NLP tools), or the annotations might be “found” (e.g. star ratings of reviews). The bare term “annotation” often refers to manual annotation.

Supervised vs. Unsupervised

$$f : x \mapsto y$$

- Supervised: observe pairs (x, y)
- Unsupervised: observe only x
- Semi-supervised: observe both

How does the data relate to the task? We can view a task as wanting to find a function f that maps from an input x to an output y.

For example, for sentiment analysis, the input would be a text, and the output would be the sentiment (positive/negative). For topic classification, the input would be a text, and the output would be a topic. For machine translation, the input would be a sentence in the source language, and the output would be a sentence in the target language.

For supervised learning, we observe both inputs and outputs. For unsupervised learning, we observe only inputs, and the task is to find some kind of structure in the data – for a probabilistic model, this might mean estimating the probability of the input; for a non-probabilistic model, this might mean grouping the input into clusters. Further reading on unsupervised learning: Ghahramani (2004) `http://mlg.eng.cam.ac.uk/zoubin/papers/ul.pdf`

Models

$$f : x \mapsto y$$

- How do we represent f?
- Parameters

There are many ways that we could represent a function. One way to do this is to define a model with a fixed structure (or architecture), but which includes many parameters which will be decided based on the data. If we specify the model and the parameters, we have a specific function. If we only specify the model, we have a family of functions.

(I will not be discussing nonparametric models, which can also be seen as having an infinite number of parameters. Further reading on nonparametric models: Gershman and Blei (2011) `https://arxiv.org/pdf/1106.2697`)

Discriminative vs. Generative

$$f : x \mapsto y$$

- Non-probabilistic: f
- Discriminative: $P(y|x)$
- Generative: $P(x, y)$

A non-probabilistic model deterministically maps inputs to outputs.

Probabilistic models can be broken down into two types. Discriminative models model the conditional distribution over outputs, given an input. Generative models model the joint probability of inputs and outputs. Generative models are more general, since $P(x, y) = P(y|x)P(x)$, so they also tell us the probability of inputs $P(x)$ – but this can also make them more challenging.

What is Machine Learning?

- Task – what function do we want?
- Data – what do we observe?
- Model – how do we represent the function?
- Training – how do we fix the representation, based on what we observe?

We can see a model as parametrising a family of functions. Training optimises the parameters, according to the observed data.

Topic Classification

- Task
  - Input: text
  - Output: topic (out of small set)
- Data
  - Texts, each labelled with a topic
  - (If unsupervised: topic *discovery*)

Topic classification is a simple example of a supervised learning task.

The related unsupervised task (where texts are unlabelled) is topic discovery, which is relevant for information retrieval and in the digital humanities.

Naive Bayes

- Generative model – $P(x, y)$

Naive Bayes

$$\underset{y}{\operatorname{argmax}}\, P(y|x) \overset{\text{Bayes}}{=} \underset{y}{\operatorname{argmax}}\, P(y)P(x|y) \quad \text{Naive}$$

$$\approx \underset{y}{\operatorname{argmax}}\, P(y)\prod_i P(x_i|y)$$

- Bernoulli NB – x_i binary-valued
- Multinomial NB – x_i integer-valued

Naive Bayes is a simple generative model, which makes a strong (“naive”) independence assumption – all input features are assumed to be independent.

For text, we can take the features to be words. In Bernoulli Naive Bayes, the features are binary-valued (1 if a word appears, 0 if not). In Multinomial Naive Bayes, they are integer-valued (the number of times the word appears) – a “bag of words”. The data must be pre-processed appropriately.

Naive Bayes can generate the training data – we first generate the topic y, and then we generate the features given the topic. For Bernoulli Naive Bayes, we independently generate each binary feature. For Multinomial Naive Bayes, we independently generate one word at a time.

The independence assumption is clearly wrong (e.g. “Hong” and “Kong”), but Naive Bayes is often surprisingly accurate. Fast and dumb models are sometimes better than slow and sophisticated models!

Naive Bayes

- Parameters: $P(y)$, $P(x_i|y)$
- Training (Bernoulli NB):
  - $P(y) = \dfrac{N_y + \alpha}{N + K\alpha}$
  - $P(x_i|y) = \dfrac{N_{y,i} + \beta}{N_y + 2\beta}$
- Hyperparameters: α, β

Bernoulli Naive Bayes, trained based on observed counts:
N – total number of documents
N_y – number of documents with topic y
$N_{y,i}$ – number of documents with topic y where word i appears
K – number of topics
More precisely, $P(x_i\!=\!1|y) = \frac{N_{y,i}+\beta}{N_y+2\beta}$, and $P(x_i\!=\!0|y) = \frac{N_y-N_{y,i}+\beta}{N_y+2\beta}$.

Multinomial Naive Bayes can be trained similarly, except:
$N_{y,i}$ – number of times word i appears in topic y documents
V – vocabulary size
Generate word i with probability $\frac{N_{y,i}+\beta}{\sum_j N_{y,j}+V\beta}$

Smoothing avoids zero probabilities (and can improve the probability estimates in general). The amount of smoothing is an example of a hyperparameter – a quantity controlling how the parameters are fixed.

Example: English Wikipedia

Thus, what started as an effort to translate between languages evolved into an entire discipline devoted to understanding how to represent and process natural languages using computers.

What topic is this about?

Which features (words) are relevant?

Example: English Wikipedia

An extreme example is the alien species, the Vulcans, who had a violent past but learned to control their emotions.

Example: German Wikipedia

Es umschließt die Mündungen des Hudson River und des East River in den Atlantischen Ozean und erhebt sich durchschnittlich sechs Meter über den Meeresspiegel.

If you speak English but not German, you might be able to guess a topic without understanding the whole text.

For many NLP tasks, full text understanding is not necessary!

Example: German Wikipedia

Schließlich bediente sich Ian Fleming auch der Geschichten und des Charakters des serbischen Doppelagenten Duško Popov aus dem Zweiten Weltkrieg.

Even a very small number of features can be enough.

Evaluation

How do we know if it works?

Training and Testing

- Split data:
  - Training
  - Development
  - Testing
- Metric (e.g. accuracy, F1)
- Baseline, significance test

Splitting the data allows us to test if a system really works. The training and test data should not overlap, and the test data should ideally only be used once. Development data is used for preliminary tests, and for hyperparameter tuning.

Cross-validation uses multiple data splits (and it should ideally only be done once).

A metric is necessary to quantitatively test a system. The results can be compared against a baseline (a simple model) and a ceiling (perfect or human performance). Significance tests can be used to probe whether a difference between systems could be a real effect, or just down to random chance (sadly often missing or done badly in NLP!)

Shared Tasks

$$\left.\begin{array}{l}\blacksquare\ \text{Task}\\ \blacksquare\ \text{Data}\end{array}\right\}\text{Provided}$$

$$\left.\begin{array}{l}\blacksquare\ \text{Model}\\ \blacksquare\ \text{Training}\end{array}\right\}\text{Participant}$$

A shared task is a standard dataset, which allows different research groups to more easily compare models or training algorithms.

Summary

- ML – task, data, model, training
- Topic classification with Naive Bayes
- Evaluation – data split, shared tasks