Lecture 13: Text classification

Trevor Cohn
(Slide credits: William Webber)

COMP90042, 2015, Semester 1
What we’ll learn in this lecture

- The classification process
- A simple text classification method tied closely to vector-space model:
  - $k$ nearest neighbours
- The logistic regression classifier, a.k.a., maximum entropy
- How to evaluate classification systems
Regression estimate real output variable for doc

Ranking rank docs by some quality

Classification assign class to doc

- Binary (two-class) classification; typically
  - predict a score for each doc
  - threshold the score → classification
- Can formulate $C$ multi-class problem as
  - predict a score for each doc, class pair
  - select top scoring class $\text{argmax}_c \text{score}(c, d)$
  - scoring function can be trained as several binary classifiers
  - or trained jointly
Classification: outline

Types of classification

- **Rule-based**  Human writes rules, machine applies
- **Decision tree**  Machine learns (discrete) rules
- **Statistical**  Machine learns statistical models

Statistical ML for classification

- Human labels example objects with classes (training data)
- Machine learns statistical model from examples
- Machine predicts class of unlabelled objects from model
$k$ nearest-neighbours

- Predicted class of object $d$
- ...majority class of $k$ training objects “nearest” $d$
- Cosine distance a possible “nearness” metric for docs
$k$ nearest-neighbours

Pros

- Good effectiveness for text
- Handles multi-class directly
- Doesn’t require model to be built
- Handles any concept of “similar”
- *Bayes optimal* in the limit of infinite data
$k$ nearest-neighbours

**Cons**

- Need to tune selection of $k$ ($\approx 40$ for text)
- Need to adjust for unbalanced classes
- Can be tricky to define similarity over complex objects (document title, first paragraph, metadata)
- **Computationally intensive** at classification time
  - $O(n)$ for naive method (compare each item)
  - $O(\log n)$ for divide-and-conquer methods
What if we need not just the class, but probabilities?

Naive Bayes is one solution

... but performs poorly with non-independent features

Logistic Regression

A.k.a. Maximum Entropy classifier, softmax neural network

- Model log-odds of classification as a linear model

\[
\log \frac{P(c = 1|d)}{P(c = 0|d)} = \vec{w}^T \phi(d)
\]

- Model weights \( \vec{w} \) learned in training

- Predefined feature representation \( \phi(d) \), e.g., vector of TF*IDF values for each term
Logistic Regression

- Rearranging, we get

\[
P(c = 1 | d) = \frac{1}{1 + \exp(-\mathbf{w}^\top \phi(d))} = \sigma \left( \mathbf{w}^\top \phi(d) \right)
\]

- \( \sigma(z) = \frac{1}{1 + \exp(-z)} \) known as the logistic sigmoid function
- Squashing function, mapping from \( \mathbb{R} \to [0, 1] \)
- Gives rise to name, as it combines real valued regression with logistic function
Logistic Regression: Training

- Trained using maximum likelihood estimate

\[
\arg\max_{\vec{w}} P(\text{train}) = \arg\max_{\vec{w}} \log P(\text{train})
\]

\[
= \arg\max_{\vec{w}} \sum_{i=1}^{N} \log P(c_i|d_i)
\]

- Unlike simple multinomial models already encountered, there is no analytic solution

- Trained using iterative gradient-based methods; initial guess of \(\vec{w} = \vec{0}\) then update in direction

\[
\frac{\partial}{\partial \vec{w}} \sum_{i=1}^{N} \log P(c_i|d_i)
\]

- Objective is convex \(\rightarrow\) will find globally optimal weights
Regularisation in Logistic Regression

Which is the better fit?

Tendency to sharpen probabilities to 0 and 1; add penalty term to training, e.g., $|\vec{w}|^2$ which limits over-fitting.

▶ a form of smoothing, like we’ve seen in other probabilistic models
Logistic Regression Summary

- Allows use of rich feature representation
  - can be highly inter-dependent
  - can support millions
  - no need to define similarity function
- Provides probability estimates
- Straight-forward extension to multi-class setting
- Faster at classification time than $k$NN; but slower to train
- Generally $k$NN, LR and SVMs all yield excellent performance

Software
See e.g., LIBLINEAR
http://www.csie.ntu.edu.tw/~cjlin/liblinear/ (there are many other implementations out there.)
Classification: outline (bis)

- Human labels example objects with classes (training data)
- Machine learns statistical model from examples
- Machine predicts class of unlabelled objects from model

Here focus on document classification, e.g.,

- classifying topic
- classifying spam or not spam
- classifying the author, their gender, age etc
- classifying the language
- classifying by the type of question it might answer (what/who/where...)
- classifying relevance to a query, or relevance ranking
Classifier: labelling

- User identifies classes $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$
- User finds, or system samples, training documents $\mathcal{T}$
- User labels each document $d \in \mathcal{T}$ with its class
- Output is set $\mathcal{T}_c$ of training examples for each class $c$
Classifier: features

Require calculable representation of objects to be classified

- Identify set of discrete *features*
- Each object represented as a *feature vector* (denoted $\phi(d)$ earlier)
  - each cell represents a feature
  - value of cell is object’s weight for that feature
- Result is an object $\times$ feature matrix
  (called the *design matrix* in statistics)
Learning algorithm

- Machine learner learns \textit{model}
  - Of class \( c \) from training examples \( T_c \)
  - Or of overall classification decision (esp. multi-class)
- A model is a function that:
  - Takes a feature vector as input
  - Produces strength of membership to each class \( c \in \mathcal{C} \)
  - Can read of argmax class assignment
- Models can work by:
  - Similarity (\( k \)NN)
  - Explicit modelling formula (LR, SVM, Naive Bayes, linear regression)
Features in text classification

For text classification:

- Objects are documents
- Terms are features
- Weights are (e.g. TF*IDF) weights

Text, compared to other forms of classification:

- Very large feature set (“for free”)
  - Feature design big issue elsewhere (e.g. image recognition)
- Highly correlated
  - Naive Bayes works poorly without feature selection
- Sparse (most features have 0 weight for most objects)
Enhancing the feature space

- Can add further document aspects as features:
  - Words in the title
  - Author, length, date of document
  - Sender, recipient of email
  - Noun phrases or n-grams
  - Number of punctuation marks, etc. etc.

- Enhancing features a “value add” for specialist applications

(Rough) decreasing order of importance for good classifier:

1. More training data
2. Better features
3. Better classification algorithm
Evaluation of (text) classification

- Classifier tested against labelled datasets
  - Dataset should be fully labelled
  - Often re-use set created by real-world process
- Classifier trained against one set of docs
- Then asked to predict labels of another set
  - Training and test set must be kept separate!
- Effectiveness measured by accuracy of prediction
Set-based Evaluation metrics

<table>
<thead>
<tr>
<th>Label</th>
<th>True</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted</td>
<td>1</td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>FN</td>
<td>TN</td>
</tr>
</tbody>
</table>

\[
\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}
\]

\[
\text{F1 score} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}
\]

\[
\text{Sensitivity (TPR, Recall)} = \frac{TP}{TP + FN}
\]

\[
\text{Specificity (TNR)} = \frac{TN}{FP + FN}
\]
Set-based evaluation metrics

- Accuracy is sensitive to imbalanced classes
  - If 95% objects in class $c$, always guessing class $c$ gets 95% accuracy
- F1 score (harmonic mean of recall and precision)
  - Also an IR metric
  - More robust to imbalance
- Multi-class evaluation uses averages over binary evaluations, per class
  - **micro-average** Sum up TP, FP, FN, TN for each class; then compute P, R, F1
  - **macro-average** Compute per-class P, R; average these and then compute F1
- Sensitivity and specificity generally used as ingredients in rank metrics (see next)
Rank metrics

- Binary classification often a “A” vs. “not-A” task
  - E.g. “about sports” vs. “not about sports”
  - I.e. “relevant” vs. “not relevant” to sports
- Many classifiers give real-valued prediction
- Can rank by decreasing association to class A
  - Cutoff point may be selected for binarization
- Ranking can be independently evaluated:
  - To evaluated quality of ranking (vs. of cutoff)
  - Because ranking might be end product
Rank metrics

- General IR rank metrics (e.g. AP) can be used
- Common alternative to graph contrasting measures down ranking
  - e.g. TPR vs FPR (sensitivity vs. 1 – specificity) at increasing ranks
- Then calculate “area under curve” (AUC) to give single measure
  - Area under TPR vs. FPR curve, referred to as the receiver operating characteristic (ROC) curve
RCV1-v2

<table>
<thead>
<tr>
<th>CCAT</th>
<th>Corporate/Industrial</th>
</tr>
</thead>
<tbody>
<tr>
<td>C11</td>
<td>Strategy/Plans</td>
</tr>
<tr>
<td>C15</td>
<td>Performance</td>
</tr>
<tr>
<td>C151</td>
<td>Accounts / Earnings</td>
</tr>
<tr>
<td>C1511</td>
<td>Annual Results</td>
</tr>
<tr>
<td>C152</td>
<td>Comment / Forecasts</td>
</tr>
</tbody>
</table>

**Figure**: Some RCV1v2 categories

- 800k-odd Reuters news articles
- 103 topical labels, manually assigned by Reuters curators
- Topics arranged in hierarchy
- One document can be labelled with more than one topic

See `nltk.corpus.reuters` for an earlier release of the corpus.
Looking back and forward

Classification process: train, learn, predict
- $k$NN simple VSM classifier
- ... follow directly from VSM search and clustering
- Maximum entropy classifier (logistic regression)
- Set-based and ranking-based classifier evaluation
Looking back and forward

Forward

- Deeper text analysis: Parts-of-speech
- Modelling the syntactic functions of words
- Sequence tagging of words in a sentence
Further reading

- Lewis, Yang, Rose, and Li, “RCV1: A New Benchmark Collection for Text Categorization Research” (JMLR, 2004) (describes the RCV1v2 collection; also gives comparative scores for kNN, Rocchio, and SVM)