Document Filtering

Summary

Document filtering - the ability to categorise a document as one of many known classes - is a classic data mining problem. In this lecture, we'll review the fundamentals of supervised machine learning and then look in detail at statistical classification techniques in which we compute and combine probabilities of features which can be measured to occur in documents.

Key points

Classification 101

- Classification is the process of assigning a class label to an object (typically represented by a vector in a feature space).
    - A supervised machine-learning algorithm uses a set of pre-labelled training data to learn how to assign class labels to vectors (and the corresponding objects).
    - A binary classifier only has two classes; a multiclass classifier has many classes.
- A linear classifier tries to learn a hyperplane that separates the feature space in two (it's a binary classifier), such that all the training points of one class lie on one side of the plane, and all the points of the other class lie on the other side.
    - Practically speaking, there are an infinite number of possible hyperplanes. In the real world, the data might not be completely linearly separable, so algorithms for learning a hyperplane try to minimise error in some respect.
    - The Perceptron algorithm is perhaps the simplest linear classifier
- Another common technique for learning a binary classifier is the Support Vector Machine (SVM).
    - A standard SVM is a linear classifier, just like the Perceptron (albeit producing a different hyperplane).
    - SVMs are maximum margin classifiers - they essentially try to position the hyperplane such that points in the two classes are as far from the hyperplane as possible
    - SVMs have the property that they can easily be made nonlinear, and create decision boundaries that are not flat planes.
- Intuitively, the simplest form of multiclass supervised classification technique is K-nearest-neighbours (KNN).
    - The class of an unlabelled vector is determined by finding the majority class of the closest (by some measure) K training points.
    - KNN can extend to any number of classes (it is multiclass), but becomes computationally expensive when there are many training examples, or the dimensionality of the feature space is large - more on this in a future lecture.
- Multiclass problems can also be solved with ensembles of linear classifiers (or even binary non-linear classifiers)
    - One versus All (OvA)/One versus Rest (OvR)
        - one classifier per class
    - One versus One (OvO)
        - $K(K-1)/2$ classifiers

Introduction to spam filtering

- Filtering spam in emails can be posed as a binary text classification problem
- Typically for spam classification we would like to use a classification technique that is *online*
    - We want the user to be able to dynamically train the classifier over time
    - Probabilistic/statistical classification methods are a natural fit for this

Classifying spam: Naïve Bayes Classification

- Start by assuming a simplified BOW model
    - Won't count terms - will just measure presence or absence of a word
    - Each term is a "feature"
- Given a set of training documents in two classes "HAM" and "SPAM"
    - Count how many documents fall into each class
    - Count how many documents for each class contain each feature
    - Can then compute the conditional probability of a feature, *f*, given a class, *c*:
      p(f | c) = n(docs with feature f in class c) / n(docs in class c)
        - Estimates will be overly affected by lack of prior evidence due to *insufficient sample size*
        - Need a better way to estimate... use Bayesian statistics to create a prior based on an assumed initial probability:
          p_w(f | c) = (weight assumed + count p_{raw}(f | c)) / (count + weight)
            - Assumption for this is that the classification for a given feature is *binomial* with a *beta distribution* prior
    - Now want to compute the conditional probability of an entire document rather than a single word
        - If we make the assumption that the features are independent, then this is nothing more than the product of the per-feature conditionals:

$$p(d|c) = \prod_{f \in d} p(f|c)$$

    - Is the assumption of independent features valid?
        - It's certainly naïve!
        - In general, it's clearly not true - there are almost certainly words that commonly appear together
        - But, in practice it doesn't seem to matter so much
        - Does also mean that we can't directly interpret the raw probability; need to choose based on most likely class
    - The ability to compute p(d|c) isn't directly useful to determining if a document is SPAM or not; actually want to compute p(c|d)
        - This is known as the posterior probability
        - Bayes theorem lets us convert a likelihood p(d|c) to the posterior p(c|d):
            - p(c|d) = p(d|c) p(c) / p(d)
                - p(c) is the prior; the probability a randomly selected document will be in class c
                    - the prior is either estimated empirically (i.e. n(docs in class c)/n(docs)) or set assuming all classes are equally likely (an unbiased prior)
                - p(d) doesn't actually matter - it's constant for all classes
                - $\text{Posterior probability} \propto \text{Likelihood} \times \text{Prior probability}$
    - Can compute p(c|d) for all categories:

$$p(c|d) \propto p(c) \prod_{f \in d} p(f|c)$$

- Simplest solution to choosing the *best* class is to choose the one with the highest probability
    - This is the Maximum a Posteriori (MAP)
- Problem with MAP is that it assumes the cost of misclassification is equal for both classes
    - This isn't true for SPAM; cost of misclassifying HAM as SPAM is considerably higher than misclassifying SPAM as HAM
    - Better approach is to look at the ratio of probabilities and apply a threshold test:
        - e.g. if (p(SPAM|d)/p(HAM|d) > 3) then SPAM else HAM

Implementation note: small probabilities

- Multiplying many small probabilities could be disastrous with fixed-precision floating point representations
    - Underflow is highly likely to occur
- Solution is to take logs to convert multiplication to addition (recall: log(ab) = log(a) + log(b)):

$$\log(p(c|d)) \propto \log(p(c)) + \sum_{f \in d} \log(p(f|c))$$

Classifying spam: Fisher's Method

- Fisher's Method is an alternative statistical test for checking whether a set of combined independent probabilities is more of less likely than a random set of probabilities
    - allows us to ask given p(c=C|f) for all features f in a document how likely is it the class is C?
- First need to compute p(c|f)
    - Bayes' theorem to the rescue again!
        - Extended form:

$$P(A_i \mid B) = \frac{P(B \mid A_i)\, P(A_i)}{\sum_j P(B \mid A_j)\, P(A_j)}$$

    - Needs a prior p(c)
        - as before could estimate from data, or make an unbiased estimate
    - For the SPAM/HAM classifier with unbiased estimate (p(SPAM) = p(HAM) = 0.5):
        - p(c | f) = p(f | c) / (p(f | HAM) + p(f | SPAM))
- Fisher's method combines k probabilities (p-*values*), $p(c=C|f_k)$, from each test into a single test statistic, X^2:

$$X^2_{2k} \sim -2 \sum_{i=1}^{k} \ln(p(c = C|f_i))$$

    - If the p-values are independent this X^2 statistic follows a chi-squared distribution with $2k$ degrees of freedom
        - The inverse chi-squared function, $K^{-1}(\cdot, \cdot)$ can give us the probability that the hypothesis c=C is true:

$$p = K^{-1}(-2 \sum_{i=1}^{k} \ln(p(c = C|f_i)), 2k) = K^{-1}(-2 \ln(\prod_{i=1}^{k} p(c = C|f_i)), 2k)$$

- Many ways to use the outcome of Fisher's method to make a prediction.
    - Popular one is:
        - $I = (1 + p_{spam} - p_{ham}) / 2$
        - I tends to be near 1 if the document is SPAM and 0 if the document is HAM

Improved features for classification

- Feature engineering can help us make better features for filtering and classification
- Documents like emails have structure which can be used to make features
    - e.g. can use the sender email, and/or a feature based on the number of addressees
- Can also create *virtual* features from simple analysis of the text content - e.g.:
    - a feature that is present if the proportion of uppercase words exceeds a threshold
    - a feature that is present if their are large numbers of misspellings
    - ...
- N-Grams can help capture context
    - e.g. bi-grams: create features from pairs of adjacent words in the text
        - Good for capturing things like names
        - Bad because number of features explodes
- Natural Language Processing can lead to even better features
    - Part-of-speech (POS) tagging can identify the important parts of the text to create features from
        - also to create more powerful features (e.g. incorporating knowledge of the POS as well as the actual word into an individual feature)
    - Named Entity Recognition (NER) can identify powerful words or sets of words
        - Potentially same benefits as N-Grams, but with fewer features

Further Reading

- Chapter 5 of "Programming Collective Intelligence" gives a good overview of the basic concepts of statistical classification
- A Statistical Approach to the Spam Problem. Robinson. 2003 http://www.linuxjournal.com/article/6467 (http://www.linuxjournal.com/article/6467)
- https://en.wikipedia.org/wiki/Naive_Bayes_spam_filtering (https://en.wikipedia.org/wiki/Naive_Bayes_spam_filtering)
- https://en.wikipedia.org/wiki/Naive_Bayes_classifier (https://en.wikipedia.org/wiki/Naive_Bayes_classifier)
- https://en.wikipedia.org/wiki/Fisher%27s_method (https://en.wikipedia.org/wiki/Fisher%27s_method)
- https://en.wikipedia.org/wiki/N-gram (https://en.wikipedia.org/wiki/N-gram)
- https://en.wikipedia.org/wiki/Named-entity_recognition (https://en.wikipedia.org/wiki/Named-entity_recognition)