# Machine Learning and Real World Data

## Sentiment Classification

Is a standard task in NLP.

**Definitions**
- A **type** is a unique word.
- A **token** is an instance of a type.
- A **tokeniser** split text into tokens.

---

### Naive Bayes Classification

**Machine learning** is a program that adapts its behaviour after being exposed to new data: without explicit programming, and implicitly from the data alone.

**Definitions**
- **Features** are observable properties of the data.
- **Classes** are labels associated with the data (e.g. positive/negative)
- **Classification** is a function that maps features to classes.

The classifier is given a set of features $O$ and classes $c_1, ..., c_n \in C$, and gives $P(c_i|O)$ for each $c_i$.
- $O = \{w_1, w_2, ..., w_n\}$ are the words in the review.
- $C = \{\text{pos}, \text{neg}\}$

Choose the $P(c_i|O)$ with the highest probability

$$\begin{aligned} c_{NB} &= \underset{c \in C}{\text{argmax}}\ P(c|O) \\ &= \underset{c \in C}{\text{argmax}}\ \frac{P(O|c)P(c)}{P(O)} \\ &= \underset{c \in C}{\text{argmax}}\ P(O|c)P(c) \end{aligned}$$

As $P(O)$ is a constant and does not affect argmax.

$$P(O|c) = P(w_1|c) \times P(w_2|C) \times \dots \times P(w_n|c)$$

$$\begin{aligned} c_{NB} &= \underset{c \in C}{\text{argmax}}\ P(c) \prod_{i=1}^{n} P(w_i|c) \\ &= \underset{c \in C}{\text{argmax}} \left( \log P(c) + \sum_{i=1}^{n} P(w_i|c) \right) \end{aligned}$$

Summing results in less floating point precision errors than multiplying.

In training, collect information needed to calculate $P(c)$ and $P(w_i|c)$

$$P(c) = \frac{N_c}{N_{\text{total}}}$$

$$P(w_i|c) = \frac{\text{count}(w_i \text{ in } c)}{\sum_{w \in V} \text{count}(w \text{ in } c)}$$

**Laplace smoothing** prevents $P(w_i|c)$ from being zero by adding 1 to the count of each type.

$$
\begin{aligned}
P(w_i|c) &= \frac{\text{count}(w_i \text{ in } c) + 1}{\sum_{w \in V} (\text{count}(w \text{ in } c) + 1)} \\
&= \frac{\text{count}(w_i \text{ in } c) + 1}{\sum_{w \in V} \text{count}(w \text{ in } c) + |V|}
\end{aligned}
$$

---

| Name | Description |
|---|---|
| Zipf's law | $$f_w \approx \frac{k}{(r_w + \beta)^\alpha}$$<br>• $f_w$ is the frequency of the word<br>• $r_w$ is the frequency rank of the word<br>• $k$, $\alpha$ and $\beta$ are language dependent constants<br>Note: $\beta$ is the rank shift. |
| Heap's law | $$u_n = kn^\beta$$<br>• $u_n$ is the number of types (vocabulary size)<br>• $n$ is the number of tokens<br>• $\beta$ and $k$ are language dependent constants |

In Naive Bayes Classification, only seen types receive a probability estimate. Adding 1 redistributes some probability mass mass to unseen types.

---

**Significance Testing**

- The **null hypothesis**: the two result sets comes from the same distribution.
- Rejecting the null hypothesis means the observed results is unlikely to have happened by chance.

Choose a **significance level** $\alpha$, reject the null hypothesis if the probability of observing the event under the null hypothesis is less than $\alpha$.

In a **binomial distribution** $B(N, q)$

$$P(X = k) = \binom{N}{k} q^k (1 - q)^{N-k}$$

$$P(X \leq k) = \sum_{i=0}^{k} \binom{N}{i} q^i (1 - q)^{N-i}$$

A **two-tailed test** tests if the two systems performs equally well: the $\alpha$ in each tail is halved.

| | **Actual = same** | **Actual = different** |
|---|---|---|
| **Predicted = same** | Correct | Type II error:<br>• Use a more powerful test (e.g. permutation test rather than sign test)<br>• Use more data |
| **Predicted = different** | Type I error | Correct |

Significance testing cannot show two distributions are the same.

**Note**

For testing sentiment classifiers, ignoring ties will lead to the null hypothesis being incorrectly rejected. Add 0.5 to the count of positive and negative results in case of ties.

---

**Overtraining**

Overtraining is where more training makes the classifier perform worse on unseen data.

Am I overtraining?

- If you are using large amounts of new test data, not overtraining.
- If incrementally improving the classifier on the same small test data, overtraining.

Overtraining is caused by finding characteristic features of each class that are hard to generalise.

**N-fold cross-validation**

1. Split data into $N$ equal folds.
2. For each fold $X$, train on other folds, test on fold $X$ only.
3. Average all the accuracy for the final accuracy.

It is good if each splits performs equally well, calculate the variance:

$$\text{var} = \frac{1}{n} \sum_{i}^{n} (x_i - \mu)^2$$

Consider the $N$ experiments as one overall experiment.

| Cross validation | Description |
|---|---|
| Stratified cross-validation | Each split mirrors the distribution of classes in the overall data. |
| Jack-knitting | Each individual data point is a split. |
| Dependency-sensitive cross-validation | Fold in a way that known characteristics of a data are isolated (e.g. one split per genre) |

Cross-validation does not solve the problem of overtraining.

Instead, a **validation corpus** (a separate set of data not used for training or testing) can be used to

- Tweak parameters before training
- Check if training is making the system perform worse on the validation corpus (is it overtraining?)

---

## Uncertainty and Agreement

$$\overline{P_a} \text{ observed agreement} = \frac{1}{N} \sum_{i=0}^{N-1} \frac{\#\text{observed pairs of agreements on item } i}{\#\text{possible pairs}}$$

where $N$ is the number of items to be classified.

$$\overline{P_e} \text{ chance agreement} = \sum_{c \in C} P(c)^2$$

**Fleiss' Kamma**

$$\kappa = \frac{\overline{P_a} - \overline{P_e}}{1 - \overline{P_e}}$$

- $\kappa = 1$ then complete agreement
- $\kappa = 0$ then no agreement beyond what is expected by chance
- $\kappa = 0.8$ means very good agreement

**Note**

$\kappa$ can be negative.

---