Spamerkennung mit log-linearen Modellen

Implementieren Sie ein weiteres Spamerkennungs-System auf Basis von log-linearen Modellen, welches dieselben Daten wie in der letzten Übung verwendet. Es sollte wieder aus einem Trainingsprogramm und einem Anwendungsprogramm bestehen. Ihr Programm sollte so allgemein implementiert werden, dass es auch für Klassifikations-Probleme mit mehr/anderen Klassen anwendbar ist.

Hier ist Pseudocode für die Aufgabe:

```
für n Epochen
    Trainingsdaten zufällig umordnen mit random.shuffle
    für jede Mail in den Trainingsdaten
        Merkmalsvektoren für alle Klassen berechnen
        Scores für alle Klassen berechnen
        p(Klasse|Mail) für alle Klassen berechnen
        Gradient durch Addition der beobachteten Merkmalswerte und
            Subtraktion der erwarteten Merkmalswerte berechnen
        Gewichtsvektor anpassen
```

Die Programme sollen folgendermaßen aufgerufen werden:

```
python3 train.py train-dir paramfile
```

```
python3 classify.py paramfile mail-dir
```

Zur Vermeidung von **Overflow** gehen Sie am besten so vor:

- Sie berechnen die Scores $s(c)$ als Produkt von Merkmals- und Gewichtsvektor
- Sie berechnen $logZ = \log \sum_c e^{s(c)} = \text{logsumexp}_c\ s(c)$
- Sie berechnen $p(c) = \frac{e^{s(c)}}{Z} = \frac{e^{s(c)}}{e^{\log Z}} = e^{s(c) - logZ}$

Wie Sie die Funktion **logsumexp** implementieren müssen, um Overflow zu vermeiden, erklärt die englische Wikipedia-Seite zum Stichwort LogSumExp im Abschnitt *log-sum-exp trick for log-domain calculations.*

Vorüberlegungen

- Welche Merkmale verwenden Sie am besten?
- Welche Datenstrukturen verwenden Sie?
- Was speichern Sie in der Parameterdatei?
- Welche Teilaufgaben umfasst das Anwendungsprogramm?

Schicken Sie das fertige **Programm**, en optimierten Wert für die **Lernrate** sowie die Liste der für die Testdaten **ausgegebenen Klassen** an `schmid@cis.lmu.de`.