Learning k-term DNF using k-CNF

In this course, you have learnt a method for learning k-term DNF formulae by first learning a k-CNF formula and then translating it to a DNF formula (which, unfortunately, may or may not be k-DNF). In this tutorial you will implement this method and use it to learn a DNF formula on a small dataset.

Data

1. Training data (not real, but artificially created) are defined in `data.m`. The training data describe patients: age, type of prescribed glasses, whether they suffer from astigmatism and whether they have normal or reduced production of tears. We want to learn a set of rules (in the form of disjunction of conjunctions, i.e. DNF) which would decide to whom contact lenses should be prescribed (positive examples) and to whom they should not (negative examples). The classification of examples (positive ... 1, negative ... -1) is stored in variable `labels`. Attribute names are stored in variable `header`.

Self-study...

2. Study the method for learning DNF formulae using k-CNF formulae from the course materials. **Briefly:** We get a multi-set of training examples. We transform this set by adding *derived attributes* corresponding to k-clauses (i.e. clauses with length k) built from the attributes of the original dataset - in general, there are $\mathcal{O}(n^k)$ such derived attributes. Then we learn a monotone conjunction on this transformed dataset. If such a conjunction exists then we have learnt a k-CNF formula (we just substitute the clauses for the derived attributes in the monotone conjunction). After that, we can transform the k-CNF formula to DNF. If we are lucky then we get a k-term DNF formula - if we were not, then we may get a larger DNF formula (containing more terms).

Implementation...

3. Implement the learning algorithm for k-CNF formulae into function `k_cnf_learn`. You can use function `transform` which adds the derived attributes (corre-

sponding to all k-clauses) to the dataset. You can also use function `cnf2str` to convert CNF formulae to string.

4. Use your function `k_cnf_learn` and a prepared function `cnf2dnf` to create a function for learning DNF formulae using k-CNF formulae (ideally, we would like to get a k-term DNF formula but that is not always possible). You can use function `dnf2str` to convert DNF formulae to string.

Testing...

5. Apply your function for DNF on the contact-lens-prescription dataset. **Expected result:**

   (tear-prod-rate_normal **&** ¬*age_presbyopic* **&** ¬*age_pre-presbyopic* **&** *age_young)* **or** *(tear-prod-rate_normal* **&** ¬*age_presbyopic* **&** ¬*astigmatism_yes* **&** ¬*age_young* **&** *age_pre-presbyopic)* **or** *(tear-prod-rate_normal* **&** ¬*age_presbyopic* **&** *spectacle-prescrip_myope* **&** ¬*age_young* **&** *age_pre-presbyopic)* **or** *(tear-prod-rate_normal* **&** ¬*spectacle-prescrip_myope* **&** ¬*astigmatism_yes* **&** ¬*age_pre-presbyopic* **&** ¬*age_young* **&** *age_presbyopic)* **or** *(tear-prod-rate_normal* **&** *astigmatism_yes* **&** *spectacle-prescrip_myope* **&** ¬*age_pre-presbyopic* **&** ¬*age_young* **&** *age_presbyopic)*

6. How would you simplify the learnt formula using domain knowledge?