Performance of soft predicate learning on simple concepts

Lingfeng Yang

July 26, 2011

1 The predicate learning algorithm

We are testing an algorithm for learning noisy/soft predicates. Currently, we only learn conjunctions of atomic predicates from a fixed background set. The algorithm is coarse-to-fine, general-to-specific; we start with the empty conjunction and add the predicate with the highest likelihood score, then proceed iteratively, refining our hypothesis by adding a predicate and picking the highest-scoring refined hypothesis at each step.

Termination criteria can be a problem because the likelihood will always go down. Currently, we have a few stopping criteria available; stop when the likelihood falls below a threshold, stop after performing a fixed number of iteration, or stop when there are no more refinements available. For testing purposes, we would like to use the final stopping criteria, as it tells us what the entire algorithm is going to do.

The relevant parts of the source code follow:

Listing 1: Coarse-to-fine soft predicate learning

```
(define (feature-induction-gen background data current-hypothesis iteration-fx
     curr-state)
  (let* ([new-hyp-score (induce-one-step background data
      current-hypothesis)]
         [shouldstop-state (iteration-fx current-hypothesis new-hyp-score
             curr-state)]
         [shouldstop (first shouldstop-state)]
         [next-state (second shouldstop-state)])
    (cond [shouldstop (list current-hypothesis next-state)]
          [else (feature-induction-gen background data (first new-hyp-score)
              iteration-fx next-state)])))

(define (induce-one-step background data current-hypothesis)
  (let* ([refinements (get-refinements background data current-hypothesis)]
         [refinement-scores (zip refinements
                                 (map (curry data-hyp->log-likelihood data) refinements))]
         [best-refinement-score (argmax second refinement-scores)])
    best-refinement-score))


(define (get-refinements background data current-hypothesis)
  ; We need generators of predicates and indexings
  ; We take them from the data itself.

  (define (generate-predicate-applications)
    (let* ([all-vars (iota (length (first data)))]
           [all-applications (lambda (b)
                               (cond [(commutative? b) (select-k-comm (pred->arity b) all-vars)]
                                     [else (select-k (pred->arity b) all-vars)]))])
      (concatenate (map (lambda (b)
                          (map (lambda (app)
                                 (list b app)) (all-applications b)))
                        background))))

  (define (already-used? pred-idx)
    (contains? pred-idx current-hypothesis))

  (define (legal-app? pred-idx)
    (and (not (already-used? pred-idx))
         (can-apply? (first pred-idx)
                     (idx->vals (first data) (second pred-idx)))))

  (let* ([applications (generate-predicate-applications)]
         [legal-refinements (filter legal-app? applications)])
    (map (lambda (r) (cons r current-hypothesis)) legal-refinements)))

(define (idx->vals row idx)
  (map (lambda (i) (list-ref row i)) idx))

(define (data-hyp->log-likelihood data hyp)
  (define (single-log-likelihood row)
    (define (apply-one-pred pred-idx)
      (let* ([pred (first pred-idx)]
             [idx (second pred-idx)]
             [val (log (apply pred (idx->vals row idx)))])
        val))
    (apply + (map apply-one-pred hyp)))
  (apply + (map single-log-likelihood data)))
```

```

(define (argmax f xs)
  (cond [(null? xs) '()]
        [else
          (first (sort (lambda (x y)
                               (> (f x) (f y))) xs))]))
```

We use two measures of performance: how fast the likelihood falls as the algorithm runs, under different levels of noise (likelihood performance), and whether the algorithm learns ”true” predicates before others (logical performance). Each measure is a function of iteration number. In the ideal case, we would like the likelihood to take a steep dive when we refine the hypothesis with a (probabilistically/softly) inconsistent predicate, and to learn (probabilistically/softly) consistent predicates before any inconsistent ones. We find that both performance measures degrade as the variance increases, but the algorithm is able to learn the right predicates in most cases.

To determine whether a predicate is ”true” relative to the ground truth, we include a set of ground-truth facts with each example along with a background theory of how the atomic predicates interact. These are expressed as Prolog programs that suceed when the predicate is entailed by the background theory and ground-truth facts of the current concept, and fail otherwise.

The background theory:

Listing 2: Background theory

```
:- table_all.

equals(X, Y) :- equals(Y, X).
equals(X, Y) :- equals(X, Z), equals(Z, Y).

greater(X, Y) :- greater(X, Z), greater(Z, Y).
greater(X, Y) :- equals(X, Z), greater(Z, Y).
greater(X, Y) :- equals(Y, Z), greater(X, Z).

neg(X, Y) :- neg(Y, X).
neg(X, Y) :- equals(X, Z), neg(Z, Y).
neg(X, Y) :- equals(Y, Z), neg(X, Z).

offby1(X, Y) :- offby1(Y, X).
offby1(X, Y) :- equals(X, Z), offby1(Z, Y).
offby1(X, Y) :- equals(Y, Z), offby1(X, Z).

offby2(X, Y) :- offby2(Y, X).
offby2(X, Y) :- equals(X, Z), offby2(Z, Y).
offby2(X, Y) :- equals(Y, Z), offby2(X, Z).

offby3(X, Y) :- offby3(Y, X).
```

```
offby3(X, Y) :− equals(X, Z), offby3(Z, Y).
offby3(X, Y) :− equals(Y, Z), offby3(X, Z).
```

2 Concept 1: Increasing sequence

Program and ground-truth facts:

Listing 3: Concept 1

```
concept1_facts = """
greater(x1, x0).
greater(x2, x1).
greater(x3, x1).
offby1(x0, x1).
offby1(x1, x2).
offby1(x2, x3).

"""

def concept1(v=0.1):
    n = sample(range(−20, 20), 1)[0]
    g = lambda : gauss(1, v)

    x0 = gauss(n, v)
    x1 = x0 + g()
    x2 = x1 + g()
    x3 = x2 + g()

    return [x0, x1, x2, x3]
```

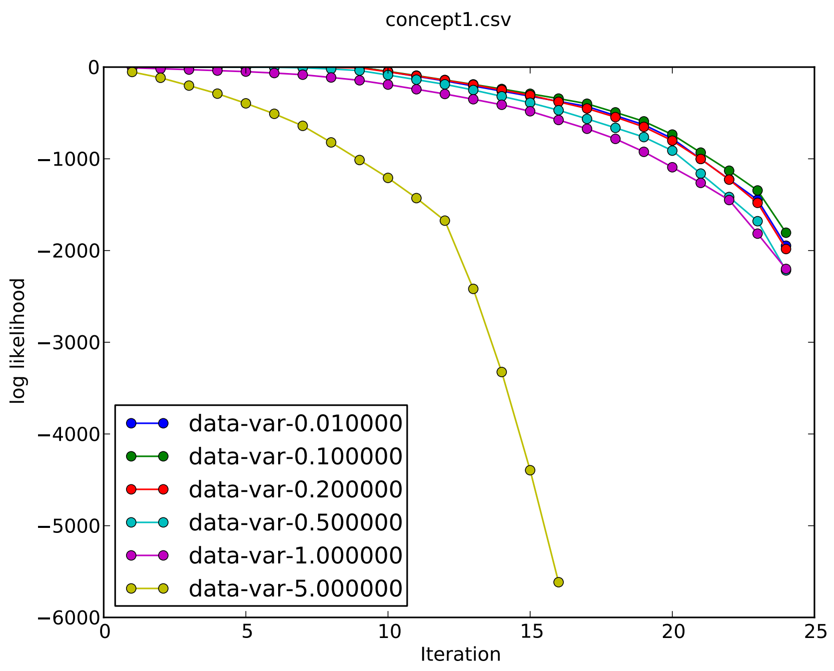
Learned predicates at each noise level:

Listing 4: Concept 1

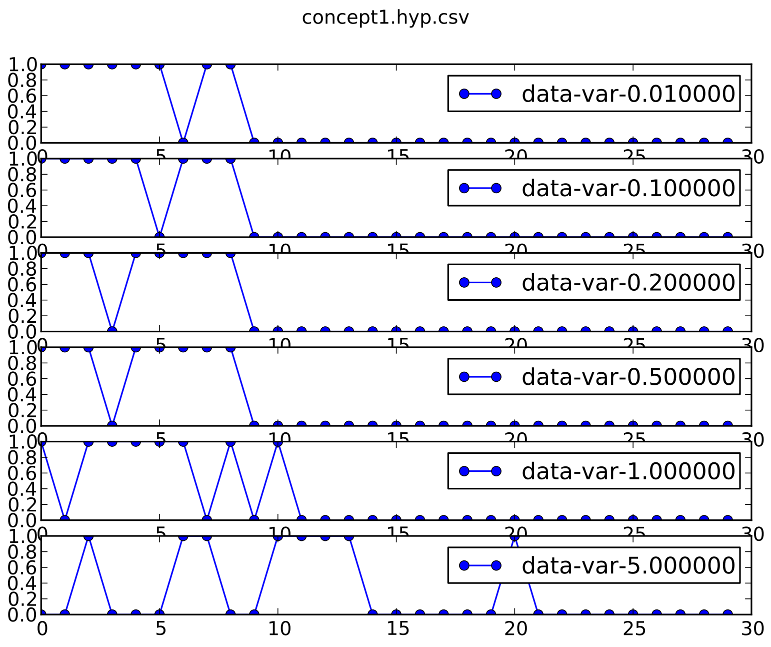
```
Name: data−var−0.010000 soft−greater? 3 0 soft−greater? 3 1 soft−greater? 2
    0 soft−offby1? 1 0 soft−offby1? 3 2 soft−offby1? 2 1 soft−greater? 3 2 soft
    −greater? 1 0 soft−greater? 2 1 soft−greater? 1 2 soft−greater? 0 1 soft−
    greater? 2 3 soft−eq? 2 1 soft−offby1? 2 0 soft−eq? 1 0 soft−offby1? 3 1
    soft−eq? 3 2 soft−greater? 0 2 soft−greater? 1 3 soft−greater? 0 3 soft−eq
    ? 2 0 soft−eq? 3 1 soft−offby1? 3 0 soft−eq? 3 0 soft−neg? 1 0 soft−neg?
    2 0 soft−neg? 3 0 soft−neg? 2 1 soft−neg? 3 1 soft−neg? 3 2

Name: data−var−0.100000 soft−greater? 3 0 soft−greater? 2 0 soft−greater? 3
    1 soft−greater? 1 0 soft−greater? 2 1 soft−greater? 3 2 soft−offby1? 2 1
    soft−offby1? 1 0 soft−offby1? 3 2 soft−offby1? 3 1 soft−greater? 2 3 soft−
    eq? 3 2 soft−greater? 1 2 soft−greater? 0 1 soft−eq? 2 1 soft−offby1? 2 0
    soft−eq? 1 0 soft−greater? 1 3 soft−greater? 0 2 soft−greater? 0 3 soft−
```

```
      offby1? 3 0 soft-eq? 3 1 soft-eq? 2 0 soft-eq? 3 0 soft-neg? 1 0 soft-neg
      ? 2 0 soft-neg? 3 0 soft-neg? 2 1 soft-neg? 3 1 soft-neg? 3 2

Name: data-var-0.200000 soft-greater? 3 0 soft-greater? 3 1 soft-greater? 2
      0 soft-greater? 3 2 soft-greater? 2 1 soft-greater? 1 0 soft-offby1? 2 1
      soft-offby1? 3 2 soft-offby1? 1 0 soft-greater? 0 1 soft-eq? 1 0 soft-
      offby1? 2 0 soft-greater? 1 2 soft-greater? 2 3 soft-eq? 2 1 soft-eq? 3 2
      soft-offby1? 3 1 soft-greater? 0 2 soft-greater? 1 3 soft-greater? 0 3 soft
      -eq? 2 0 soft-offby1? 3 0 soft-eq? 3 1 soft-eq? 3 0 soft-neg? 1 0 soft-
      neg? 2 0 soft-neg? 3 0 soft-neg? 2 1 soft-neg? 3 1 soft-neg? 3 2

Name: data-var-0.500000 soft-greater? 3 0 soft-greater? 3 1 soft-greater? 2
      0 soft-greater? 3 2 soft-greater? 2 1 soft-greater? 1 0 soft-offby1? 3 2
      soft-offby1? 2 1 soft-offby1? 1 0 soft-greater? 0 1 soft-greater? 2 3 soft-
      greater? 1 2 soft-eq? 3 2 soft-eq? 2 1 soft-eq? 1 0 soft-offby1? 3 1 soft-
      offby1? 2 0 soft-greater? 0 2 soft-greater? 1 3 soft-greater? 0 3 soft-eq?
      3 1 soft-eq? 2 0 soft-offby1? 3 0 soft-eq? 3 0 soft-neg? 1 0 soft-neg? 2
      0 soft-neg? 3 0 soft-neg? 2 1 soft-neg? 3 1 soft-neg? 3 2

Name: data-var-1.000000 soft-greater? 2 0 soft-greater? 3 2 soft-greater? 2
      1 soft-greater? 3 0 soft-greater? 3 1 soft-greater? 1 0 soft-offby1? 1 0
      soft-greater? 0 1 soft-offby1? 3 2 soft-greater? 2 3 soft-offby1? 2 1 soft-
      offby1? 2 0 soft-greater? 1 2 soft-eq? 1 0 soft-greater? 0 2 soft-eq? 3 2
      soft-greater? 1 3 soft-greater? 0 3 soft-eq? 2 1 soft-eq? 2 0 soft-offby1?
       3 0 soft-offby1? 3 1 soft-eq? 3 1 soft-eq? 3 0 soft-neg? 1 0 soft-neg? 2
       0 soft-neg? 3 0 soft-neg? 2 1 soft-neg? 3 1 soft-neg? 3 2

Name: data-var-5.000000 soft-greater? 3 2 soft-greater? 0 1 soft-greater? 2
      1 soft-greater? 1 2 soft-greater? 0 2 soft-greater? 2 3 soft-greater? 3 1
      soft-greater? 1 0 soft-greater? 0 3 soft-greater? 1 3 soft-greater? 2 0 soft
      -greater? 3 0 soft-offby1? 2 1 soft-offby1? 3 2 soft-eq? 2 1 soft-eq? 3 2
      soft-eq? 1 0 soft-eq? 2 0 soft-eq? 3 0 soft-eq? 3 1 soft-offby1? 1 0 soft
      -offby1? 2 0 soft-offby1? 3 0 soft-offby1? 3 1 soft-neg? 1 0 soft-neg? 2
      0 soft-neg? 3 0 soft-neg? 2 1 soft-neg? 3 1 soft-neg? 3 2
```

Likelihood performance:



Logical performance:



3 Concept 2: Shared increasing sequence

Program and ground-truth facts:

Listing 5: Concept 2

```
concept2_facts = """
equals(x0, x1).
equals(x2, x3).
greater(x2, x0).
offby1(x2, x0).

"""

def concept2(v = 0.1):
    n= sample(range(-20, 20), 1)[0]

    x0 = gauss(n, v)
    x1 = x0 + gauss(0, v)

    x2 = x1 + gauss(1, v)
    x3 = x2 + gauss(1, v)

    return [x0, x1, x2, x3]
```

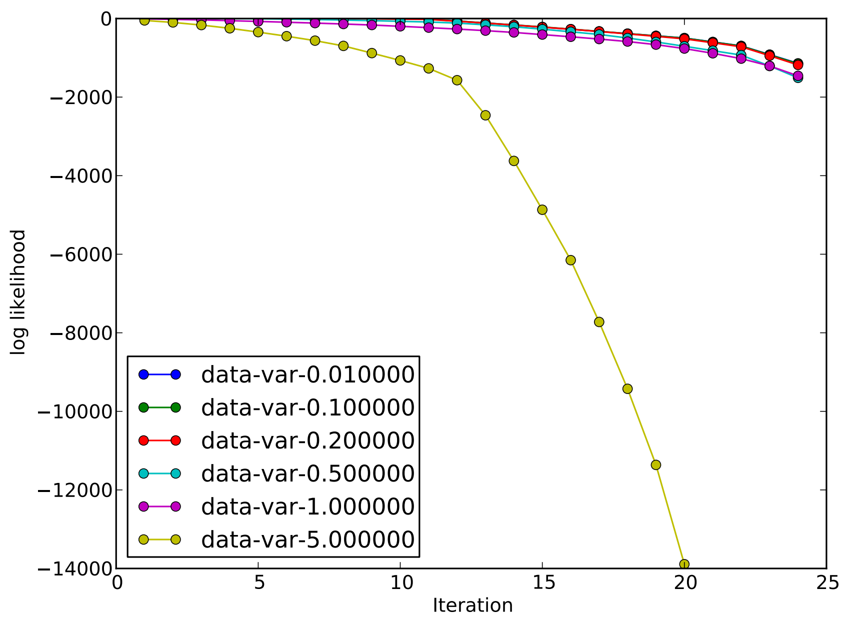
Learned predicates at each noise level:

Listing 6: Concept 2

```
Name: data-var-0.010000 soft-greater? 3 1 soft-greater? 3 0 soft-eq? 1 0
    soft-offby1? 2 1 soft-offby1? 3 2 soft-offby1? 2 0 soft-greater? 3 2 soft-
    greater? 2 1 soft-greater? 2 0 soft-greater? 0 1 soft-greater? 1 0 soft-
    greater? 0 2 soft-greater? 1 2 soft-greater? 2 3 soft-offby1? 1 0 soft-eq?
    2 0 soft-offby1? 3 0 soft-eq? 2 1 soft-offby1? 3 1 soft-eq? 3 2 soft-
    greater? 0 3 soft-greater? 1 3 soft-eq? 3 0 soft-eq? 3 1 soft-neg? 1 0 soft
    -neg? 2 0 soft-neg? 3 0 soft-neg? 2 1 soft-neg? 3 1 soft-neg? 3 2

Name: data-var-0.100000 soft-greater? 3 1 soft-greater? 3 0 soft-greater? 2
    1 soft-greater? 3 2 soft-greater? 2 0 soft-offby1? 3 2 soft-offby1? 2 1 soft
    -eq? 1 0 soft-offby1? 2 0 soft-greater? 0 1 soft-greater? 1 0 soft-offby1?
     1 0 soft-greater? 2 3 soft-greater? 0 2 soft-greater? 1 2 soft-eq? 3 2 soft
    -eq? 2 1 soft-offby1? 3 1 soft-offby1? 3 0 soft-eq? 2 0 soft-greater? 0 3
    soft-greater? 1 3 soft-eq? 3 1 soft-eq? 3 0 soft-neg? 1 0 soft-neg? 2 0
    soft-neg? 3 0 soft-neg? 2 1 soft-neg? 3 1 soft-neg? 3 2

Name: data-var-0.200000 soft-greater? 3 1 soft-greater? 3 0 soft-greater? 3
    2 soft-greater? 2 1 soft-greater? 2 0 soft-eq? 1 0 soft-offby1? 3 2 soft-
    offby1? 2 0 soft-offby1? 2 1 soft-greater? 0 1 soft-greater? 1 0 soft-
    offby1? 1 0 soft−greater? 0 2 soft−greater? 1 2 soft−greater? 2 3 soft−eq?
    2 0 soft−eq? 2 1 soft−offby1? 3 0 soft−eq? 3 2 soft−offby1? 3 1 soft−
    greater? 0 3 soft−greater? 1 3 soft−eq? 3 0 soft−eq? 3 1 soft−neg? 1 0 soft
    −neg? 2 0 soft−neg? 3 0 soft−neg? 2 1 soft−neg? 3 1 soft−neg? 3 2

Name: data−var−0.500000 soft−greater? 3 1 soft−greater? 3 2 soft−greater? 3
    0 soft−greater? 2 1 soft−greater? 2 0 soft−greater? 0 1 soft−offby1? 3 2
    soft−offby1? 2 0 soft−offby1? 2 1 soft−eq? 1 0 soft−greater? 1 0 soft−
    offby1? 1 0 soft−greater? 0 2 soft−greater? 1 2 soft−greater? 2 3 soft−eq?
    2 1 soft−eq? 2 0 soft−eq? 3 2 soft−greater? 0 3 soft−greater? 1 3 soft−
    offby1? 3 0 soft−offby1? 3 1 soft−eq? 3 0 soft−eq? 3 1 soft−neg? 1 0 soft−
    neg? 2 0 soft−neg? 3 0 soft−neg? 2 1 soft−neg? 3 1 soft−neg? 3 2

Name: data−var−1.000000 soft−greater? 3 2 soft−greater? 3 1 soft−greater? 2
    1 soft−offby1? 3 2 soft−greater? 0 1 soft−offby1? 1 0 soft−greater? 3 0 soft
    −greater? 1 0 soft−greater? 2 0 soft−greater? 1 2 soft−offby1? 2 1 soft−
    greater? 2 3 soft−greater? 0 2 soft−offby1? 3 1 soft−eq? 1 0 soft−greater?
    1 3 soft−eq? 3 2 soft−greater? 0 3 soft−eq? 2 1 soft−offby1? 2 0 soft−
    offby1? 3 0 soft−eq? 3 1 soft−eq? 2 0 soft−eq? 3 0 soft−neg? 1 0 soft−neg
    ? 2 0 soft−neg? 3 0 soft−neg? 2 1 soft−neg? 3 1 soft−neg? 3 2

Name: data−var−5.000000 soft−greater? 3 2 soft−greater? 3 1 soft−greater? 1
    0 soft−greater? 3 0 soft−greater? 2 1 soft−greater? 2 0 soft−greater? 1 2
    soft−greater? 0 1 soft−greater? 2 3 soft−greater? 0 2 soft−greater? 1 3 soft
    −greater? 0 3 soft−offby1? 1 0 soft−offby1? 2 1 soft−offby1? 3 2 soft−eq?
    1 0 soft−eq? 2 1 soft−eq? 3 2 soft−offby1? 2 0 soft−eq? 2 0 soft−eq? 3 0
    soft−eq? 3 1 soft−offby1? 3 0 soft−offby1? 3 1 soft−neg? 1 0 soft−neg? 2
    0 soft−neg? 3 0 soft−neg? 2 1 soft−neg? 3 1 soft−neg? 3 2
```

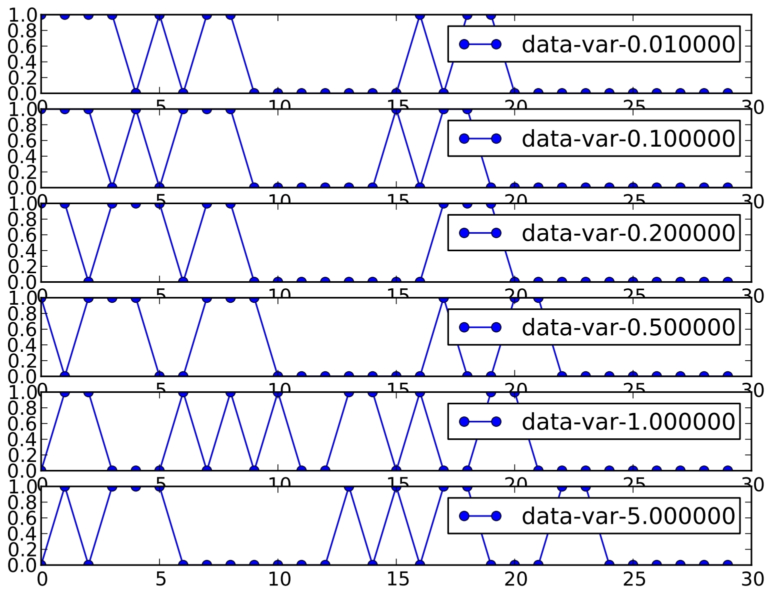
Likelihood performance:

concept2.csv



Logical performance:

concept2.hyp.csv



4 Concept 3: Negation

Program and ground-truth facts:

Listing 7: Concept 3

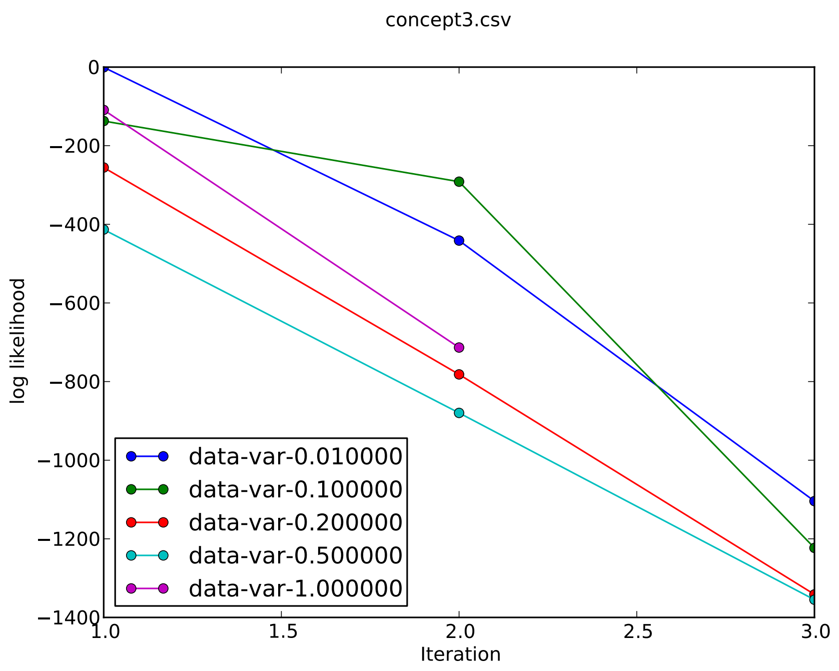
```
concept3_facts= """
neg(x0, x1).

"""

def concept3(v = 0.1):

    n = sample(range(-20, 20), 1)[0]

    x0 = gauss(n, v)
    x1 = x0 * gauss(-1, v)

    return [x0, x1]
```

Learned predicates at each noise level:

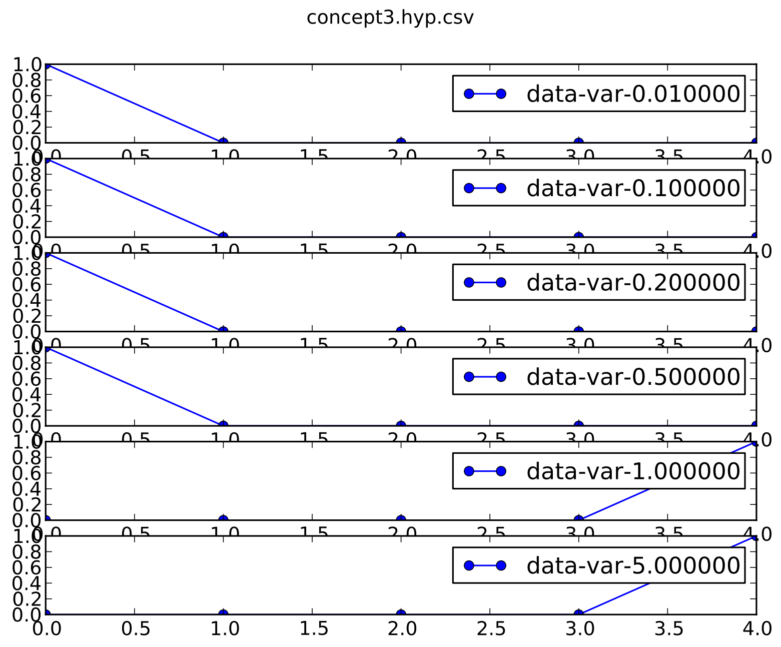
Listing 8: Concept 3

```
Name: data-var-0.010000 soft-neg? 1 0 soft-greater? 1 0 soft-greater? 0 1
    soft-eq? 1 0 soft-offby1? 1 0

Name: data-var-0.100000 soft-neg? 1 0 soft-greater? 1 0 soft-greater? 0 1
    soft-eq? 1 0 soft-offby1? 1 0

Name: data-var-0.200000 soft-neg? 1 0 soft-greater? 1 0 soft-greater? 0 1
    soft-eq? 1 0 soft-offby1? 1 0

Name: data-var-0.500000 soft-neg? 1 0 soft-greater? 1 0 soft-greater? 0 1
    soft-eq? 1 0 soft-offby1? 1 0

Name: data-var-1.000000 soft-greater? 1 0 soft-greater? 0 1 soft-eq? 1 0
    soft-offby1? 1 0 soft-neg? 1 0

Name: data-var-5.000000 soft-eq? 1 0 soft-greater? 1 0 soft-greater? 0 1
    soft-offby1? 1 0 soft-neg? 1 0
```

Likelihood performance:



Logical performance:



5 Concept 4: Multiple negation

Program and ground-truth facts:

Listing 9: Concept 4

```
concept4_facts = """
neg(x0, x2).
neg(x1, x3).

"""

def concept4(v = 0.1):
    n = lambda : sample(range(−20, 20), 1)[0]

    x0 = gauss(n(), v)
    x1 = gauss(n(), v)
    x2 = x0 ∗ gauss(−1, v)
    x3 = x1 ∗ gauss(−1, v)

    return [x0, x1, x2, x3]
```

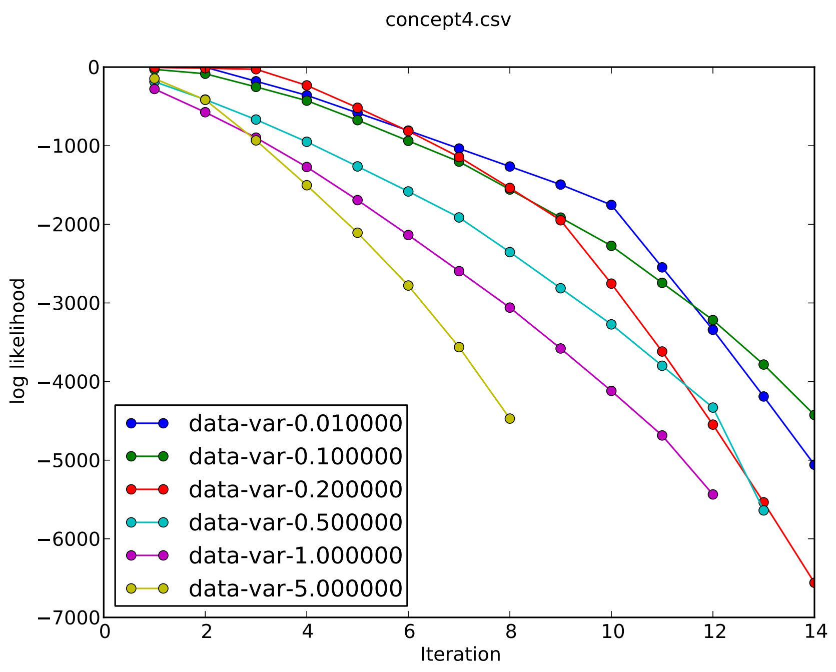
Learned predicates at each noise level:

Listing 10: Concept 4

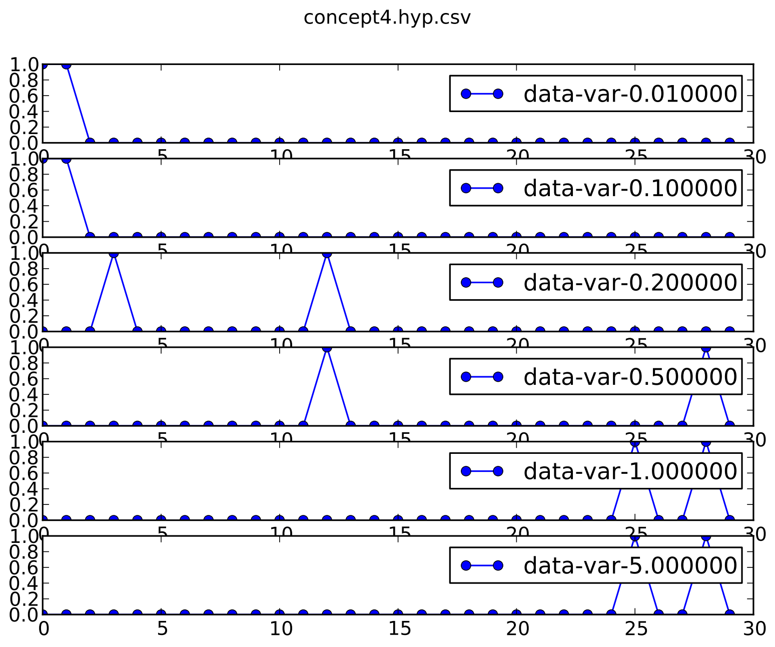
```
Name: data−var−0.010000 soft−neg? 2 0 soft−neg? 3 1 soft−greater? 3 0 soft−
    greater? 2 1 soft−greater? 2 3 soft−greater? 1 0 soft−greater? 3 2 soft−
    greater? 2 0 soft−greater? 0 1 soft−greater? 3 1 soft−greater? 0 3 soft−
    greater? 1 2 soft−greater? 0 2 soft−greater? 1 3 soft−eq? 1 0 soft−eq? 2 0
    soft−eq? 3 0 soft−eq? 2 1 soft−eq? 3 1 soft−eq? 3 2 soft−offby1? 1 0 soft
    −offby1? 2 0 soft−offby1? 3 0 soft−offby1? 2 1 soft−offby1? 3 1 soft−
    offby1? 3 2 soft−neg? 1 0 soft−neg? 3 0 soft−neg? 2 1 soft−neg? 3 2

Name: data−var−0.100000 soft−neg? 2 0 soft−neg? 3 1 soft−greater? 2 3 soft−
    greater? 1 0 soft−greater? 1 3 soft−greater? 0 3 soft−greater? 1 2 soft−
    greater? 2 1 soft−greater? 2 0 soft−greater? 3 0 soft−greater? 3 2 soft−
    greater? 0 1 soft−greater? 0 2 soft−greater? 3 1 soft−eq? 1 0 soft−eq? 2 0
    soft−eq? 3 0 soft−eq? 2 1 soft−eq? 3 1 soft−eq? 3 2 soft−offby1? 1 0 soft
    −offby1? 2 0 soft−offby1? 3 0 soft−offby1? 2 1 soft−offby1? 3 1 soft−
    offby1? 3 2 soft−neg? 1 0 soft−neg? 3 0 soft−neg? 2 1 soft−neg? 3 2

Name: data−var−0.200000 soft−greater? 1 0 soft−greater? 2 3 soft−greater? 1
    3 soft−neg? 3 1 soft−greater? 2 0 soft−greater? 0 3 soft−greater? 1 2 soft
    −greater? 2 1 soft−greater? 3 0 soft−greater? 0 1 soft−greater? 3 2 soft−
    greater? 3 1 soft−neg? 2 0 soft−greater? 0 2 soft−eq? 1 0 soft−eq? 2 0 soft
    −eq? 3 0 soft−eq? 2 1 soft−eq? 3 1 soft−eq? 3 2 soft−offby1? 1 0 soft−
    offby1? 2 0 soft−offby1? 3 0 soft−offby1? 2 1 soft−offby1? 3 1 soft−offby1?
     3 2 soft−neg? 1 0 soft−neg? 3 0 soft−neg? 2 1 soft−neg? 3 2
```

```

Name: data-var-0.500000 soft-greater? 0 3 soft-greater? 1 2 soft-greater? 3 0 soft-greater? 2 1 soft-greater? 1 0 soft-greater? 2 3 soft-greater? 1 3 soft-greater? 0 1 soft-greater? 2 0 soft-greater? 3 2 soft-greater? 3 1 soft-greater? 0 2 soft-neg? 2 0 soft-eq? 1 0 soft-eq? 2 0 soft-eq? 3 0 soft-eq? 2 1 soft-eq? 3 1 soft-eq? 3 2 soft-offby1? 1 0 soft-offby1? 2 0 soft-offby1? 3 0 soft-offby1? 2 1 soft-offby1? 3 1 soft-offby1? 3 2 soft-neg? 1 0 soft-neg? 3 0 soft-neg? 2 1 soft-neg? 3 1 soft-neg? 3 2

Name: data-var-1.000000 soft-greater? 3 0 soft-greater? 2 1 soft-greater? 2 3 soft-greater? 3 2 soft-greater? 3 1 soft-greater? 1 0 soft-greater? 2 0 soft-greater? 1 2 soft-greater? 0 3 soft-greater? 1 3 soft-greater? 0 1 soft-greater? 0 2 soft-eq? 1 0 soft-eq? 2 0 soft-eq? 3 0 soft-eq? 2 1 soft-eq? 3 1 soft-eq? 3 2 soft-offby1? 1 0 soft-offby1? 2 0 soft-offby1? 3 0 soft-offby1? 2 1 soft-offby1? 3 1 soft-offby1? 3 2 soft-neg? 1 0 soft-neg? 2 0 soft-neg? 3 0 soft-neg? 2 1 soft-neg? 3 1 soft-neg? 3 2

Name: data-var-5.000000 soft-greater? 1 0 soft-greater? 2 0 soft-greater? 2 1 soft-greater? 0 1 soft-greater? 1 2 soft-greater? 3 0 soft-greater? 0 2 soft-greater? 3 2 soft-eq? 1 0 soft-eq? 2 0 soft-eq? 3 0 soft-eq? 2 1 soft-eq? 3 1 soft-eq? 3 2 soft-greater? 3 1 soft-greater? 0 3 soft-greater? 1 3 soft-greater? 2 3 soft-offby1? 1 0 soft-offby1? 2 0 soft-offby1? 3 0 soft-offby1? 2 1 soft-offby1? 3 1 soft-offby1? 3 2 soft-neg? 1 0 soft-neg? 2 0 soft-neg? 3 0 soft-neg? 2 1 soft-neg? 3 1 soft-neg? 3 2
```

Likelihood performance:



Logical performance:



6 Concept 5: Symmetric negation

Program and ground-truth facts:

Listing 11: Concept 5

```
concept5_facts = """
neg(x0, x3).
neg(x1, x2).
offby1(x0, x1).
offby1(x2, x3).
"""

def concept5(v = 0.1):
    n = sample(range(-20, 20), 1)[0]

    x1 = gauss(n, v)
    x0 = x1 + gauss(1, v)

    x2 = x1 * gauss(-1, v)
    x3 = x0 * gauss(-1, v)

    return [x0, x1, x2, x3]
```

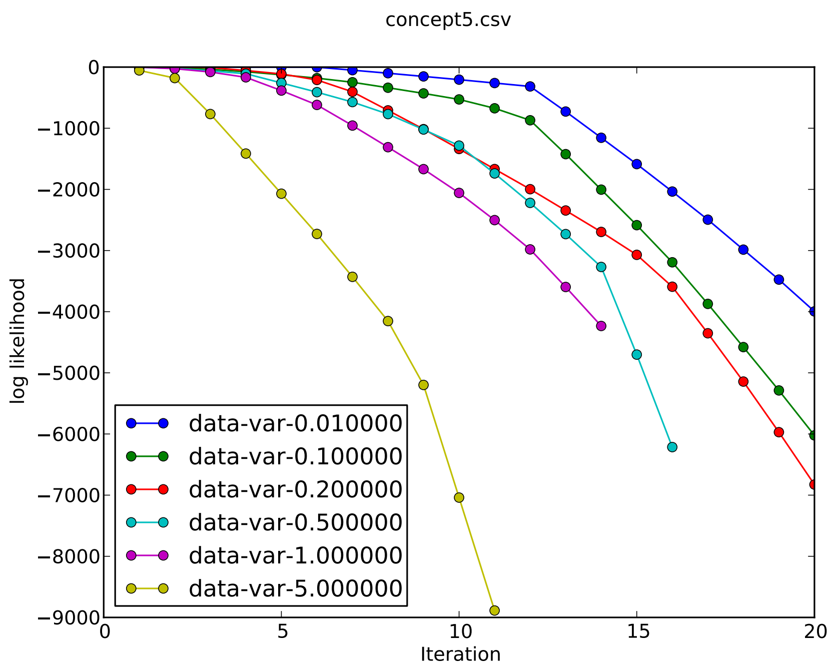
Learned predicates at each noise level:

Listing 12: Concept 5

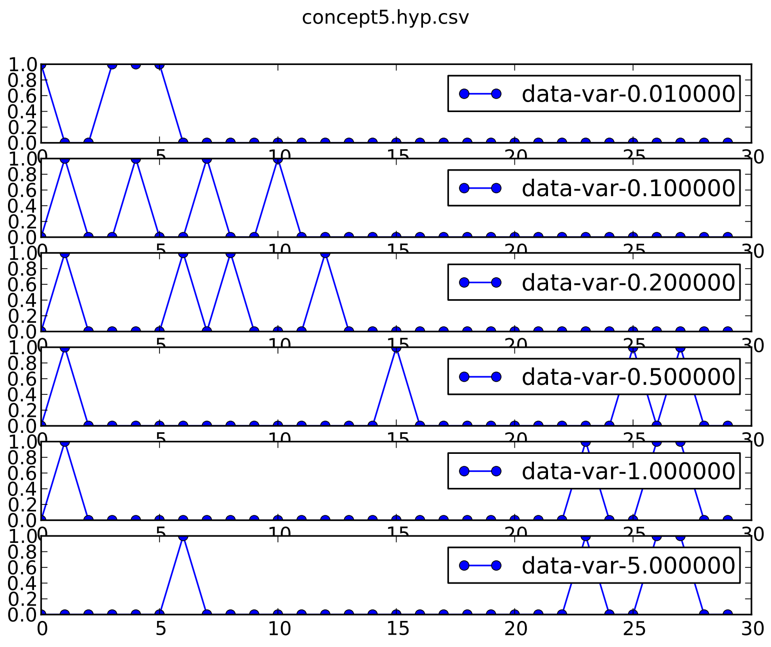
```
Name: data-var-0.010000 soft-offby1? 1 0 soft-greater? 0 1 soft-greater? 2 3
     soft-neg? 2 1 soft-neg? 3 0 soft-offby1? 3 2 soft-greater? 3 2 soft-
    greater? 1 0 soft-eq? 3 2 soft-neg? 2 0 soft-neg? 3 1 soft-eq? 1 0 soft-
    greater? 0 3 soft-greater? 0 2 soft-greater? 1 3 soft-greater? 1 2 soft-
    greater? 2 1 soft-greater? 2 0 soft-greater? 3 1 soft-greater? 3 0 soft-eq?
     2 0 soft-eq? 3 0 soft-eq? 2 1 soft-eq? 3 1 soft-offby1? 2 0 soft-offby1?
    3 0 soft-offby1? 2 1 soft-offby1? 3 1 soft-neg? 1 0 soft-neg? 3 2

Name: data-var-0.100000 soft-greater? 0 1 soft-offby1? 1 0 soft-greater? 2 3
     soft-greater? 3 2 soft-neg? 3 0 soft-greater? 1 0 soft-eq? 1 0 soft-
    offby1? 3 2 soft-neg? 2 0 soft-neg? 3 1 soft-neg? 2 1 soft-eq? 3 2 soft-
    greater? 2 1 soft-greater? 2 0 soft-greater? 3 1 soft-greater? 3 0 soft-
    greater? 0 2 soft-greater? 0 3 soft-greater? 1 2 soft-greater? 1 3 soft-eq?
     2 0 soft-eq? 3 0 soft-eq? 2 1 soft-eq? 3 1 soft-offby1? 2 0 soft-offby1?
    3 0 soft-offby1? 2 1 soft-offby1? 3 1 soft-neg? 1 0 soft-neg? 3 2

Name: data-var-0.200000 soft-greater? 0 1 soft-offby1? 1 0 soft-greater? 2 3
     soft-greater? 1 0 soft-eq? 1 0 soft-greater? 3 2 soft-neg? 2 1 soft-
    greater? 2 1 soft-neg? 3 0 soft-greater? 2 0 soft-neg? 3 1 soft-greater? 3
     1 soft-offby1? 3 2 soft-greater? 3 0 soft-neg? 2 0 soft-eq? 3 2 soft-
```

```
greater? 0 3 soft-greater? 1 3 soft-greater? 0 2 soft-greater? 1 2 soft-eq?
 2 0 soft-eq? 3 0 soft-eq? 2 1 soft-eq? 3 1 soft-offby1? 2 0 soft-offby1?
3 0 soft-offby1? 2 1 soft-offby1? 3 1 soft-neg? 1 0 soft-neg? 3 2

Name: data-var-0.500000 soft-greater? 0 1 soft-offby1? 1 0 soft-greater? 1 0
 soft-eq? 1 0 soft-greater? 3 1 soft-greater? 3 2 soft-greater? 3 0 soft-
greater? 2 3 soft-greater? 2 1 soft-greater? 2 0 soft-greater? 0 3 soft-
greater? 1 3 soft-greater? 0 2 soft-greater? 1 2 soft-neg? 2 0 soft-neg? 2
 1 soft-eq? 2 0 soft-eq? 3 0 soft-eq? 2 1 soft-eq? 3 1 soft-eq? 3 2 soft-
offby1? 2 0 soft-offby1? 3 0 soft-offby1? 2 1 soft-offby1? 3 1 soft-offby1?
 3 2 soft-neg? 1 0 soft-neg? 3 0 soft-neg? 3 1 soft-neg? 3 2

Name: data-var-1.000000 soft-greater? 0 1 soft-offby1? 1 0 soft-greater? 1 0
 soft-eq? 1 0 soft-greater? 0 3 soft-greater? 1 3 soft-greater? 3 2 soft-
greater? 2 3 soft-greater? 0 2 soft-greater? 1 2 soft-greater? 3 1 soft-
greater? 3 0 soft-greater? 2 1 soft-greater? 2 0 soft-eq? 2 0 soft-eq? 3 0
soft-eq? 2 1 soft-eq? 3 1 soft-eq? 3 2 soft-offby1? 2 0 soft-offby1? 3 0
soft-offby1? 2 1 soft-offby1? 3 1 soft-offby1? 3 2 soft-neg? 1 0 soft-neg
? 2 0 soft-neg? 3 0 soft-neg? 2 1 soft-neg? 3 1 soft-neg? 3 2

Name: data-var-5.000000 soft-greater? 0 1 soft-greater? 1 0 soft-greater? 0
3 soft-greater? 2 3 soft-greater? 1 3 soft-greater? 2 1 soft-offby1? 1 0
soft-greater? 2 0 soft-eq? 1 0 soft-greater? 0 2 soft-greater? 1 2 soft-eq
? 2 0 soft-eq? 3 0 soft-eq? 2 1 soft-eq? 3 1 soft-eq? 3 2 soft-greater? 3
0 soft-greater? 3 1 soft-greater? 3 2 soft-offby1? 2 0 soft-offby1? 3 0 soft
-offby1? 2 1 soft-offby1? 3 1 soft-offby1? 3 2 soft-neg? 1 0 soft-neg? 2
0 soft-neg? 3 0 soft-neg? 2 1 soft-neg? 3 1 soft-neg? 3 2
```

Likelihood performance:



Logical performance:


concept5.hyp.csv
data-var-0.010000
data-var-0.100000
data-var-0.200000
data-var-0.500000
data-var-1.000000
data-var-5.000000