

000	055
001	056
002	057
003	058
004	059
005	060
006	061
007	062
008	063
009	064
010	065
011	066
012	067
013	068
014	069
015	070
016	071
017	072
018	073
019	074
020	075
021	076
022	077
023	078
024	079
025	080
026	081
027	082
028	083
029	084
030	085
031	086
032	087
033	088
034	089
035	090
036	091
037	092
038	093
039	094
040	095
041	096
042	097
043	098
044	099
045	100
046	101
047	102
048	103
049	104
050	105
051	106
052	107
053	108
054	109

---

## Latent Gaussian Processes for Distribution Estimation of Multivariate Categorical Data – Appendix

---

## A Appendix

### A.1 Code

The basic model and inference (without covariance matrix caching) can be implemented in 20 lines of Python and Theano for each categorical variable:

```

1 | import theano.tensor as T
2 | m = T.dmatrix('m') # ..and other variables
3 | X = m + s * randn(N, Q)
4 | U = mu + L.dot(randn(M, K))
5 | Kmm = RBF(sf2, l, Z)
6 | Kmn = RBF(sf2, l, Z, X)
7 | Knn = RBFnn(sf2, l, X)
8 | KmmInv = T.matrix_inverse(Kmm)
9 | A = KmmInv.dot(Kmn)
10 | B = Knn - T.sum(Kmn * KmmInv.dot(Kmn), 0)
11 | F = A.T.dot(U)+B[:,None]**0.5 * randn(N,K)
12 | S = T.nnet.softmax(F)
13 | KL_U, KL_X = get_KL_U(), get_KL_X()
14 | LS = T.sum(T.log(T.sum(Y * S, 1)))
15 | - KL_U - KL_X
16 | LS_func = theano.function([''inputs''],
17 |                             LS)
18 | dLS_dm = theano.function([''inputs''],
19 |                             T.grad(LS, m)) # and others
20 | # ... and optimise LS with RMS-PROP

```