50.039 – Theory and Practice of Deep Learning

Alex

Week 02: Pytorch for a start

1 Task1

The goal is to let you see the power of broadcasting for speeding up computations. Also to see that you can use pytorch with GPU to speed up other computations than old deep learning.

You have seen in the linear regression lecture an RBF kernel which is a matrix

$$\phi(x_i, t_j) = \exp(-\frac{\|X[i,:] - T[j,:]\|^2}{\gamma})$$

Inside is a l2 distance matrix between $X[i,:]$ and $T[j,:]$.

$$X.size() = (N, D)$$
$$T.size() = (P, D)$$

X are features with dimensionality D and sample size N. T are prototypes with dimensionality D and sample size P. Use pytorch to compute

$$\phi(x_i, t_j) = \exp(-\frac{\|X[i,:] - T[j,:]\|^2}{\gamma})$$

the rbf-kernel matrix in pytorch.
Approach:

- decompose formula into a sequence of computations
- how to compute a term $\|X(i,:) - T(j,:)\|^2$ in pytorch ?
- how to reshape X, T such that you can use broadcasting to get $d_{ij} = \|X(i,:) - T(j,:)\|^2$?

Compare time measurements:

- two for-loops over i, j (for x_i, t_j)
- numpy broadcasting (`np.exp` and so on)
- pytorch cpu
- optional: pytorch on a gpu (a cheap notebook gpu suffices) for N, P, D nicely large

Validate that pytorch cpu gives you the same result as one of the two: for loops or numpy broadcasting.

Note if $T = X$, then this is a RBF kernel which is popular with SVMs for small data problems.