50.021 – AI

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 aged 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 $T[j,:]$, this is actually a so-called RBF kernel, shapes are:

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

Here we go for computing another kernel, which makes sense over histogram features. In numpy we can represent the set of all features $x_i, i = 1, \dots, n$ of a dataset as a numpy array such that $x_i = X[i,:]$.

Suppose x_i is a histogram feature that is $X[i,:] \geq 0$, and $\sum_d X[i,d] = 1$

Use pytorch to compute

$$k(x_i, z_j) = \sum_d \min(x_i^{(d)}, z_j^{(d)})$$

the hik-kernel matrix in pytorch. X are features from one dataset with dimensionality D and sample size N. Z are features from another dataset with dimensionality D and sample size L.
Approach:

- decompose the formula into a sequence of computations
- how to reshape X, Z such that you can use broadcasting to get $\min(x_i^{(d)}, z_j^{(d)})$?

Compare time measurements:

- two for-loops over i, j (for x_i, t_j)
- numpy broadcasting
- 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.
What data to use ? Think how you can randomly draw histograms.