50.039 – Theory and Practice of Deep Learning

Alex

Week 01: Linear and ridge regression

[The following notes are compiled from various sources such as textbooks, lecture materials, Web resources and are shared for academic purposes only, intended for use by students registered for a specific course. In the interest of brevity, every source is not cited. The compiler of these notes gratefully acknowledges all such sources.]

1 Task1

The goal is to let you do a bit recap on how to use numpy functions and arrays. Work in `linreg_studentversion.py`

You have to fill two implementations. First is:

```
def datagen2d(w1,w2,eps,num):
```

Implement in this function a data generator for 2d features with regression ground truth labels as follows:

The function returns a 2d numpy array of features x with shape $(num, dims)$ where num is the number of samples, dims is 2 here. The features are drawn from a standard normal distribution.

The labels are computed from the features as

$$y_i = x_{i,0}w_1 + x_{i,1}w_2 + n$$
$$n \sim N(0, \epsilon)$$

```
def linreg_train(xtr,ytr,C):
```

here xtr are the train features, ytr are the train labels and C is the stabilization constant from the ridge regression.

implement here the algorithm which returns the weights w of the linear regression predictor $y = x \cdot w$ according to the explicit solution obtained in the lecture.

Test your code by the `run1(···)` function

2 Task2

The goal is to see the power of feature transformations. Work in `linreg3_studentversion.py`

If you run this file, then you see at first data generated from a 1-dimensional noisy cosine. Obviously linear regression cannot do much with it.
Now implement an RBF kernel in

```
def rbffeats(x,protos,gamma):
```

In order to get a good result, with γ being in a suitable value range, implement it as

$$\exp(-\frac{dists}{\gamma})$$

The good question is why this works with an rbf kernel to approximate the cosine well? In the equation

$$y = wx + b$$

w is a slope. The solution is to view the inner product

$$w \cdot \phi(x) = \sum_d w_d \phi_d(x)$$

as a sum of regression slopes w_d weighted by a similarity $\phi_d(x)$ of data point x to prototypes indexed by $_d$. For a good choice of γ, only a few of the weights $\phi_d(x)$ will be large, and all others near zero. Thus only very few regression slopes w_d will be active for every data point x and all others are masked out!!