AIML - CS 335

Lab 9: More on Kernel

06 October 2022

Lecturer: Abir De

Important: Please read the instructions mentioned in the questions carefully. We have provided boilerplate code for each question. Please ensure that you make changes in the areas marked with TODO. Please read the comments in the code carefully for detailed information regarding input and output format.

1 Kernel SVM Regression

Support Vector Machines can be utilized to solve regression problems as well as classification problems. In the previous lab, we have seen how the kernel trick allows us to train linear models to fit non-linearly separable data. In a similar vein, the inclusion of the kernel trick in SVM-based regression allows us to train non-linear regression models that can perform well when dealing with complex datasets.

In this question, you have to train SVM regression models for two datasets:

1. **Dataset 1**: This is a simple 2-D dataset, where the x value represents the feature variable and the y value represents the target variable. The boilerplate code includes a function to plot the fitted regression model against the training data for your reference.
2. **Dataset 2**: This is a more complex dataset with a feature vector $\mathbf{x}$ of length ten for each data sample. The target variable y is a scalar value.

For each of the above datasets, you have to train three different regression models. Each regression model uses a different kernel function. The three kernel functions that you have to implement are:

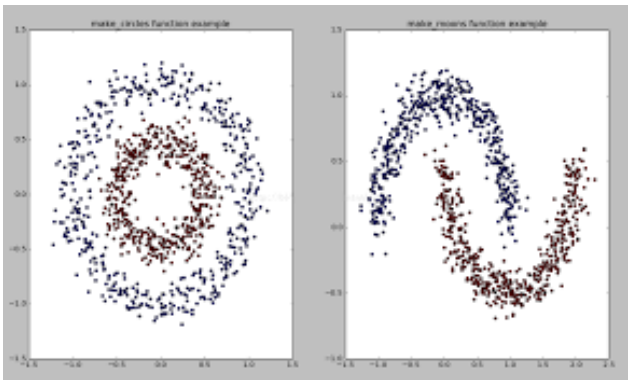
1. **Gaussian kernel:** Mathematically, it is defined as given below, with one tunable hyper-parameter σ:
$$k(a, b) = \exp\left(-\frac{||a - b||^2}{2\sigma^2}\right)$$
2. **Gaussian radial basis function (RBF):** Mathematically, it is defined as given below, with one tunable hyper-parameter γ:
$$k(a, b) = \exp(-\gamma||a - b||^2)$$
3. **Laplace RBF kernel:** Mathematically, it is defined as given below, with one tunable hyper-parameter σ:
$$k(a, b) = \exp\left(-\frac{||a - b||}{\sigma}\right)$$

Train the required regression models while tuning the hyper-parameters for each kernel.

2 Kernel PCA

PCA(Principal Component Analysis) is a very important tool in data analysis and machine learning. PCA finds a set of basis vectors (or directions) along which variation of data is maximum and such that all the vectors are orthogonal and ranked according to the variance of data along them. The direction with highest variance means it is most important in capturing the difference in data points. PCA is also helpful in dimensionality reduction. This is done by discarding the projects of points along directions where variance is minimal.

But PCA is a linear method. It can only capture variation along the linear dimension. Sometimes we want to capture variation in data along a non-linear dimension. This is where Kernel PCA comes in handy.



In datasets like those above, Kernel PCA can capture variations along non-linear dimensions without manually projecting the data onto a higher dimensional space. Using an appropriate kernel can also help make the data linearly separable by taking projections of the principle components in higher dimensional space.

You need to complete a function wherein, given a dataset with 2D points and a kernel name and you will need to implement kernel PCA and return projections of the the points along the top two PCA vectors in the high dimensional space.

The kernel argument to the function can take values

- **'rbf'** : RBF kernel, $K(x_1, x_2) = e^{-\gamma||x-y||^2}$ where $\gamma = 15$ for this problem
- **'poly'** : Polynomial kernel, $K(x_1, x_2) = (1 + x_1^T x_2)^d$ where $d = 5$ for this problem
- **'radial'** : Custom kernel with $K(x_1, x_2) = r_{x_1} r_{x_2} + \theta_{x_1} \theta_{x_2}$ where (r_{x_i}, θ_{x_1}) is the projection of x_i in the radial basis

Refer **https://people.eecs.berkeley.edu/ wainwrig/stat241b/scholkopf_kernel.pdf** for more information on kernel PCA

3 Non-parametric Regression

We know Linear Regression setting where we have to find relationship between two continuous variables x and y. Given a data in the form of N feature vector x = $[x_1, x_2, ...x_n]$ and the corresponding feature vector y, we try to find a line that best fit the data, by estimating weight vector w.

Now consider the scenario where data doesn't fit in a straight line, linear regression is unlikely to fit in. For this we can use polynomial regression. But we cannot know the degree of polynomial up to which we should go.

Here comes the Kernel Regression to rescue! Unlike Linear or Polynomial regression where we need to estimate weight vector w, Kernel regression is non-parametric, meaning it estimates target y_i by performing computations directly on input x_i

Given data points (x_i, y_i) kernel regression predicts by first computing a kernel k for each data points x_i. Then for a given test input x_t, it computes a similarity score with each x_i (given by $x_i - x_t$) using kernel; the similarity(s_i) score acts as a weight w_i that represents the importance of that kernel (and corresponding label y_i) in predicting target y_t.

$$s_i = \frac{N * k(\frac{x_i - x_t}{b})}{\sum_{i=1}^{N} k(\frac{x_i - x_t}{b})}$$

The prediction is then obtained by multiplying the weight vector s = $[s_1, s_2, ..., s_n]$ with label vector y = $[y_1, y_2, ..., y_n]$.

$$y_t = \frac{1}{N} \sum_{i=1}^{N} w_i * y_i$$

Now, there can be different kernel functions which give rise to different types of kernel regression. In this question you have to implement Gaussian Kernel Regression. Here each constructed kernel can also be viewed as a normal distribution with mean value x_i and standard deviation b. The equation for the Gaussian kernel k is given below.

$$k(z) = \frac{1}{\sqrt{2\pi}} \exp(\frac{-z^2}{2})$$

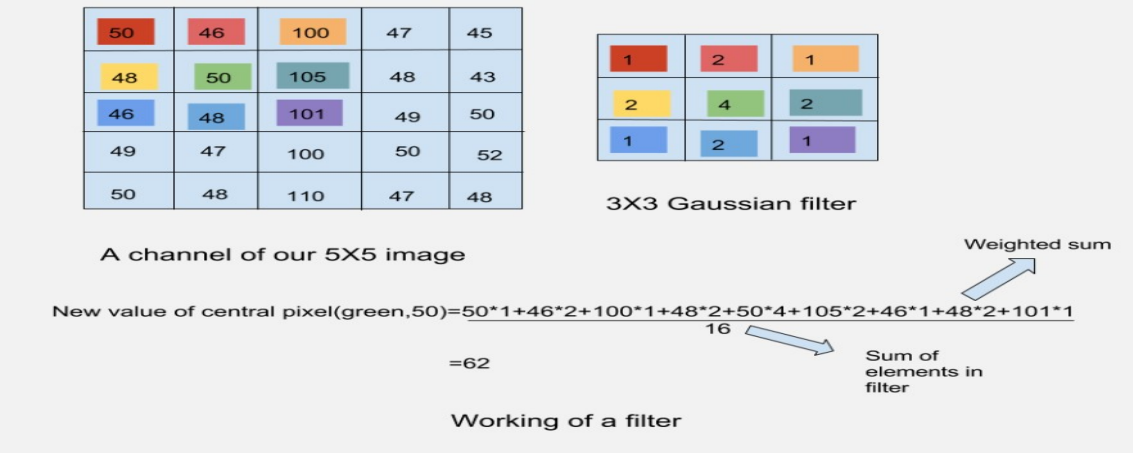
Your task is to implement Kernel Regression as explained above

4 Gradient Filtering

There are multiple convolutional filters which are used in (CNNs) to extract features from images like sobel filters can perform an edge detection as discussed in previous lab.
Similarly,**Gaussian filtering** is used to blur images and remove noise and detailing.Gaussian kernel coefficients are sampled from the 2D Gaussian function.

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Where σ is the standard deviation of the distribution, Larger values of σ produce a wider peak (greater blurring)



- The central pixel(50,green) is modified with the obtained mean value(62).
- Similarly this operation is carried out by moving the filter.
- the kernel has odd number of rows and columns so that it always has a central element.

.Consider the Following Example,

15	**20**	**25**	25	15	10
20	**15**	**50**	30	20	15
20	**50**	**55**	60	30	20
20	15	65	30	15	30
15	20	30	20	25	30
20	25	15	20	10	15

when this original image is applied on the following Gaussian kernel with mean equal to 1.

0.0625	0.125	0.0625
0.125	0.25	0.125
0.0625	0.125	0.0625

we get the following resultant Feature Map

15	20	25	25	15	10
20	29	50	35	20	15
20	50	55	60	30	20
20	31	65	37	15	30
15	20	30	20	25	30
20	25	15	20	10	15

Complete the **GaussianFilter(x, w,stride)** Function which returns the resultant image.

5 Submission instructions

Make changes only in the places mentioned in comments. Do not modify anything else. Finally, place the 3 folders, namely `Q1`, `Q2`,`Q3` and `Q4`inside a folder named `<ROLL_NUMBER>_L9` and compress it to a tar file named `<ROLL_NUMBER>_L9.tar.gz` using the command
`tar -zcvf <ROLL_NUMBER>_L9.tar.gz <ROLL_NUMBER>_L9`
Submit the tar file on Moodle. The directory structure should be -

```
<ROLL_NUMBER>_L9
| - - - - Q1
| - - - - -|- - - - Q1.py
| - - - - -|- - - - data
| - - - - -|- - - - utils.py
| - - - - Q2
| - - - - -|- - - - Q2.py
| - - - - Q3
| - - - - -|- - - - Q3.py
| - - - - -|- - - - data
| - - - - -|- - - - utils.py
| - - - - Q4
| - - - - -|- - - - Q4.py
```

Replace `ROLL_NUMBER` with your own roll number. If your Roll number has alphabets, they should be in *"small"* letters.