# ML_HW4_Q1_NS

April 9, 2021

## 1 Question 1

### 1.1 Kernel Ridge Regression

#### 1.1.1 a)

Apply kernelized ridge regression to the automobile mpg dataset. The training data and test data are provided in auto mpg train.csv and auto mpg test.csv, respectively. The first column is the mpg data while the other 7 columns are the features: 1. mpg: continuous 2. cylinders: multi-valued discrete 3. displacement: continuous 4. horsepower: continuous 5. weight: continuous 6. acceleration: continuous 7. model year: multi-valued discrete 8. origin: multi-valued discrete

We have normalized the feature data to the range [0,1]. Please apply the kernelized ridge regression to this dataset (use mpg as the target, the other 7 columns as features). Report the RMSE (Root Mean Square Error) of the models on the test data. Try (set lamda = 1).

[26]:
```
#imports
import pandas as pd
import numpy as np
from sklearn.kernel_ridge import KernelRidge
from sklearn.metrics import accuracy_score, mean_squared_error
```

[27]:
```
#we need to normlize the features between 0 and 1, this is the function to do␣
 ↪so

def NormalizeData(data):
    return (data - np.min(data)) / (np.max(data) - np.min(data))
```

[28]:
```
#read in train and test data

te = pd.read_csv('auto_mpg_test.csv', sep = ' ')

tr = pd.read_csv('auto_mpg_train.csv', sep = ' ')


tes = pd.DataFrame(te).to_numpy()

tra = pd.DataFrame(tr).to_numpy()

#split features (x) and labels (mpg, y)
```

```
x_tra = tra[:, 1:8]

y_train = tra[:, 0]

x_tes = tes[:, 1:8]

y_test = tes[:, 0]

#normalize features (x_test and x_train)

x_test = NormalizeData(x_tes)

x_train = NormalizeData(x_tra)

print(x_train.shape, y_train.shape, x_test.shape, y_test.shape)
```

```
(299, 7) (299,) (97, 7) (97,)
```

Once we can you sklearn (THANK YOU!!!!!), we can use the built in sklearn.kernel_ridge.KernelRidge function.

Thankfully, we can call the KernelRidge function with the kernel='rbf' instead of it's default linear kernel.

The rbf kernel uses the gaussian distribution asked in the question. See this link for documentation proof: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.rbf_kernel.html The default rbf kernel sets sigma = 1, as asked in the question.

[29]:
```
#Kernel Ridge Regression from sklearn using rbf (gaussian) kernel
skKRR = KernelRidge(alpha=1.0, kernel='rbf', gamma = 1/2)

#fit kernel to train data
skKRR.fit(x_train, y_train)

#predict test data
skKRR_y_pred = skKRR.predict(x_test)

#report root mean squared error
rmse = np.sqrt(mean_squared_error(y_true=y_test,y_pred=skKRR_y_pred))
print("Root Mean Squared Error for Gaussian Kernel: ",rmse)
```

```
Root Mean Squared Error for Gaussian Kernel:  3.2102263635542707
```

Out of curiosity (and because this is a short question due to sklearn), I want to try to see the difference between linear and rbf kernels.

[30]:
```
#Kernel Ridge Regression from sklearn using liner kernel
skKRR_lin = KernelRidge(alpha=1.0)
```

```python
#fit kernel to train data
skKRR_lin.fit(x_train, y_train)

#predict test data
skKRR_y_pred_lin = skKRR_lin.predict(x_test)

#report root mean squared error
rmse_lin = np.sqrt(mean_squared_error(y_true=y_test,y_pred=skKRR_y_pred_lin))
print("Root Mean Squared Error for Liner Kernel: ",rmse_lin)
```

```
Root Mean Squared Error for Liner Kernel:  5.207784717373817
```

Looks like the Gaussian Kernel is superior!

Q2) Principal Component Analysis

(a) We denote $h_j = \sum_{k=1}^{K} \| W_k^{(j)} \|^2$ i.e., the square of $L_2$ norm of the j-th row vector in $W$.

Prove that $0 \leq h_j \leq 1$ and $\sum_{j=1}^{D} h_j = k$

(i) Prove that $0 \leq h_j \leq 1$

Each column of $W_k^{(j)}$ is an orthonormal basis vector. of $\in \mathbb{R}^{D \times k}$.

Once $h_j$ is a sum of $L_2$ norms, $h_j \geq 0$ must be true, negative values would be positive due to squareing in the $L_2$ norm

Due to the $L_2$ norms of columns being always 1, due to the orthonormal nature of those columns, the $L_2$ norms will always be $\leq 1$

$\boxed{\text{Therefore, } 0 \leq h_j \leq 1}$

QED

(ii) Prove $\sum_{j=1}^{D} h_j = k$

$W_n = U\vec{a}_n$, therefore, $W = \begin{bmatrix} \vec{u}_1^T \vec{a}_1 & \cdots & \vec{u}_1 \vec{a}_n \\ \vdots & & \vdots \\ \vec{u}_D a_1 & \cdots & u_D \vec{a}_n \end{bmatrix}$

If we take the sum of the $L_2$ norms of each column vector $W$, we will have a bunch of 1's as solutions to the $L_2$ norm of each colmn vector (due to them being orthonormal basis vectors). We then have k columns in $W$, we will therefore have k number of 1's sumed up to get $\sum_{j=1}^{D} h_j = k$.

QED

(b) Prove that:

$$\max_{\substack{W \\ W^TW=I_K}} \sum_{K=1}^{K} w_K^T \Lambda w_K = \max_{\substack{h_j \\ W^TW=I_K}} \sum_{j=1}^{D} h_j \lambda_j$$

We start with $\max_{\substack{W \\ W^TW=I_K}} \sum_{K=1}^{K} w_K^T \Lambda w_K$

The definition of $w_K = U^T \vec{a}_K$ can be used, also $w_K^T = \vec{a}_K^T U$

we substitute to get:

$$\max_{\substack{W \\ W^TW=I_K}} \sum_{K=1}^{K} \vec{a}_K^T U \Lambda U^T \vec{a}_K$$

We can then use the identity $U \Lambda U^T = \sum_{j=1}^{D} \lambda_j \vec{U}_j \vec{U}_j^T$ to get

$$\max_{\substack{W \\ W^TW=I_K}} \sum_{K=1}^{K} \sum_{j=1}^{D} \vec{a}_K^T \lambda_j \vec{U}_j \vec{U}_j^T \vec{a}_K \Rightarrow \max_{\substack{W \\ W^TW=I_K}} \sum_{K=1}^{K} \sum_{j=1}^{D} \lambda_j (a_K^T \vec{U}_j)(a_j^T \vec{a}_K)^T$$

Then by the definition of $L^2$ norm we have

$$\max_{\substack{W \\ W^TW=I_K}} \sum_{K=1}^{K} \sum_{j=1}^{D} \lambda_j \|a_K^T \vec{u}_j\|^2$$ by the definiton of $W_K$ we have:
(L2 norm $W_K^T$ = L2 norm $W_K$)

$$\max_{\substack{W \\ W^TW=I_K}} \sum_{K=1}^{K} \sum_{j=1}^{D} \lambda_j \|w_K^{(j)}\|^2$$ we can flip the sammations because order does not matter

$$\max_{\substack{W \\ W^TW=I_K}} \sum_{j=1}^{D} \lambda_j \sum_{K=1}^{K} \|w_K^{(j)}\|^2$$ recall the desinition of $h_j = \sum_{K=1}^{K} \|w_K^{(j)}\|^2$

$$\boxed{\max_{\substack{h_j \\ W^TW=I_K}} \sum_{j=1}^{D} h_j \lambda_j}$$

QED

(c) What are the optimal $h_j$ in (3)? Show that $a_K = u_K$ $(k=1,\dots,K)$ is a sol of (3)

(i) What are the optimal $h_j$ in (3)?

An optimal solution would be setting $a_K$ to be an eigenvector (a column of $U$ that holds all eigenvectors (aka $a_K = u_K$). There will be a 1 with a bunch of zeros that will give us a sum of 1.
There will be $\lambda_j$'s that are 1 and 0, We want to choose where the values are $\boxed{1}$.

(ii) Show that $a_K = u_K$ is a solution of 3.

Lets start with:

$$\max_{\substack{W \\ W^TW=I_K}} \sum_{k=1}^{K} w_k^T \Delta w_k$$ Then once $w_k = U^T \bar{a}_n$ ; $w_k^T = \bar{a}_n^T U$

$$\Rightarrow \max_{\substack{W \\ W^TW=I_K}} \sum_{k=1}^{n} \bar{a}_n^T U \Delta U^T \bar{a}_n$$ We then sub our $a_n = u_k$

$$\Rightarrow \max_{\substack{W \\ W^TW=I_K}} \sum_{k=1}^{K} u_k^T U \Delta U^T \bar{u}_k$$ from part b, we can show it's equivalent to

$$\Rightarrow \max_{\substack{W \\ W^TW=I_K}} \sum_{k=1}^{n} \sum_{j=1}^{D} \lambda_j \|u_k^T u_j\|^2$$

So $U^T \bar{u}_k$ will have one pair come out to 1, and the rest zero.

We will have ones across the diagonal for K rows, the rest will be zeros. We then have it multiplied by the diagonal matrix $\Delta$, this will give us $\lambda_j$ that from $\lambda_1 \to \lambda_K$

We will then get $\lambda$ for each multiplication when $a_K = u_K$ vesus getting a partial $\lambda$ when we have just $a_K$. This $\lambda$ will be an optimal solution.