

Homework #1

Due on February 11th, 2020

Machine Learning
CS559WS—Spring 2020
Professor In Suk Jang

Daniel Kadyrov
10455680

Problem 1

By using a change of variables, verify that the univariate Gaussian distribution given by:

$$N(x | \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x - \mu)^2\right\}$$

satisfies $E(x) = \mu$. Next, by differentiating both sides of normalization condition

$$\int_{-\infty}^{\infty} N(x | \mu, \sigma^2) dx = 1$$

with respect to σ^2 , verify that the Gaussian satisfies $E(x^2) = \mu^2 + \sigma^2$.

Solution

$$\begin{aligned} E(x) &= \int_{-\infty}^{\infty} p(x|\mu, \sigma^2) x dx = E(x) = \int_{-\infty}^{\infty} \frac{x}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\} dx \\ E(x) &= \int_{-\infty}^{\infty} \frac{x-\mu}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{x^2}{2\sigma^2}\right\} dx = \int_{-\infty}^{\infty} \frac{x}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{x^2}{2\sigma^2}\right\} dx - \int_{-\infty}^{\infty} \frac{\mu}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{x^2}{2\sigma^2}\right\} dx \end{aligned}$$

The first part of the integral:

$$\begin{aligned} & - \int_0^{-\infty} x \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{x^2}{2\sigma^2}\right\} dx + \int_0^{\infty} x \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{x^2}{2\sigma^2}\right\} dx = \\ & \int_0^{\infty} (-x) \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{(-x)^2}{2\sigma^2}\right\} dx + \int_0^{\infty} x \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{x^2}{2\sigma^2}\right\} dx = \\ & - \int_0^{\infty} x \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{x^2}{2\sigma^2}\right\} dx + \int_0^{\infty} x \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{x^2}{2\sigma^2}\right\} dx = 0 \end{aligned}$$

So now the integral is:

$$E(x) = \int_{-\infty}^{\infty} \mu \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{x^2}{2\sigma^2}\right\} dx = \frac{2\mu}{\sqrt{\pi}} \int_0^{\infty} e^{-x^2} dx$$

$$E(x) = \lim_{i \rightarrow \infty} \frac{2\mu}{\sqrt{\pi}} \int_0^i e^{-x^2} dx = \mu \lim_{i \rightarrow \infty} \text{erf}(i) = \mu \Leftarrow$$

$$E(x) = \mu \rightarrow E(x)^2 = \mu^2$$

$$E(x^2) - E(x)^2 = E(x^2) - \mu^2 = \sigma^2$$

$$E(x^2) = \mu^2 + \sigma^2 \Leftarrow$$

Problem 2

Use $E(x) = \mu$ to prove $E(xx^T) = \mu\mu^T + \Sigma$. Now, using the results two definitions, show that:

$$E[x_n x_m] = \mu\mu^T + I_{nm}\Sigma$$

where x_n denotes a data point same from a Gaussian distribution with mean μ and covariance Σ , and I_{nm} denotes the (n, m) element of the identity matrix. Hence prove the result (2.124)

Solution

$$\begin{aligned}\Sigma &= E[(x - E[x])(x - E[x])^T] \\ &= E[xx^T - xE[x]^T - E[x]x^T + E[x]E[x]^T] \\ &= E[xx^T - x\mu^T - \mu x^T + \mu\mu^T] \\ &= E[xx^T] - \mu\mu^T - \mu\mu^T + \mu\mu^T \\ &= E[xx^T] - \mu\mu^T \\ E[xx^T] &= \mu\mu^T + \Sigma \Leftarrow\end{aligned}$$

Problem 3

Consider a linear model of the form:

$$f(x, w) = w_0 + \sum_{i=1}^D w_i x_i$$

together with a sum of squares/loss function of the form:

$$L_D(w) = \frac{1}{2} \sum_{n=1}^N (f(x_n, w) - y_n)^2$$

Now suppose that Gaussian noise ϵ_i with zero mean and variance σ^2 is added independently to each of the input variables x_i . By making use of $E[\epsilon_i] = 0$ and $E[\epsilon_i \epsilon_j] = \delta_{ij} \sigma^2$, show that minimizing L_D averaged over the noise distribution is equivalent to minimizing the sum of square error for noise-free input variables with the addition of a weight-decay regularization term, in which the bias parameters w_0 is omitted from the regularizer.

Solution

$$\begin{aligned} y_n(x_n, w) &= w_0 + \sum_{i=1}^D w_i (x_{ni} + \epsilon_{ni}) = w_0 + \sum_{i=1}^D w_i x_{ni} + \sum_{i=1}^D w_i \epsilon_{ni} = y(x_n, w) + \sum_{i=1}^D w_i \epsilon_{ni} \\ E_{dn}(w) &= \frac{1}{2} \sum_{n=1}^N (y_n(x_n, w) - t_n)^2 = \frac{1}{2} (y(x_n, w) + \sum_{i=1}^D w_i \epsilon_{ni} - t_n)^2 \\ &= \frac{1}{2} \sum_{n=1}^N ((y(x_n, w) - t_n)^2 + 2(y(x_n, w) - t_n) (\sum_{i=1}^D w_i \epsilon_{ni}) + (\sum_{i=1}^D w_i \epsilon_{ni})^2) \\ \mathbb{E}(L_{Dn}(w)) &= \frac{1}{2} \sum_{n=1}^N ((y(x_n, w) - t_n)^2 + 2(y(x_n, w) - t_n) (\sum_{i=1}^D w_i \mathbb{E}(\epsilon_{ni})) + \mathbb{E}((\sum_{i=1}^D w_i \epsilon_{ni})^2)) \end{aligned}$$

$\mathbb{E}(\epsilon_{ni}) = 0$ so:

$$\begin{aligned} \mathbb{E}((\sum_{i=1}^D w_i \epsilon_{ni})^2) &= \mathbb{E}(\sum_{i_1=1}^D \sum_{i_2=1}^D w_{i_1} w_{i_2} \epsilon_{ni_1} \epsilon_{ni_2}) \\ &= \sum_{i_1=1}^D \sum_{i_2=1}^D w_{i_1} w_{i_2} \mathbb{E}(\epsilon_{ni_1} \epsilon_{ni_2}) \\ &= \sum_{i_1=1}^D \sum_{i_2=1}^D w_{i_1} w_{i_2} \delta_{i_1 i_2} \\ &= \sum_{i=1}^D w_i^2 \\ \mathbb{E}(L_{Dn}(w)) &= \frac{1}{2} \sum_{n=1}^N ((y(x_n, w) - t_n)^2 + \sum_{i=1}^D w_i^2) = L_D(w) + \frac{N}{2} \sum_{i=1}^D w_i^2 \Leftarrow \end{aligned}$$

Problem 4

UCI Machine Learning: Bike Sharing Data Set

Build at least four regression models (e.g., linear, polynomial, non-linear) to predict the count of total rental bikes including both casual and registered. Explore data to reduce the number of features. Use K-fold cross validation and report the mean squared error (MSE) on the testing data. You need to write down every step in your experiment.

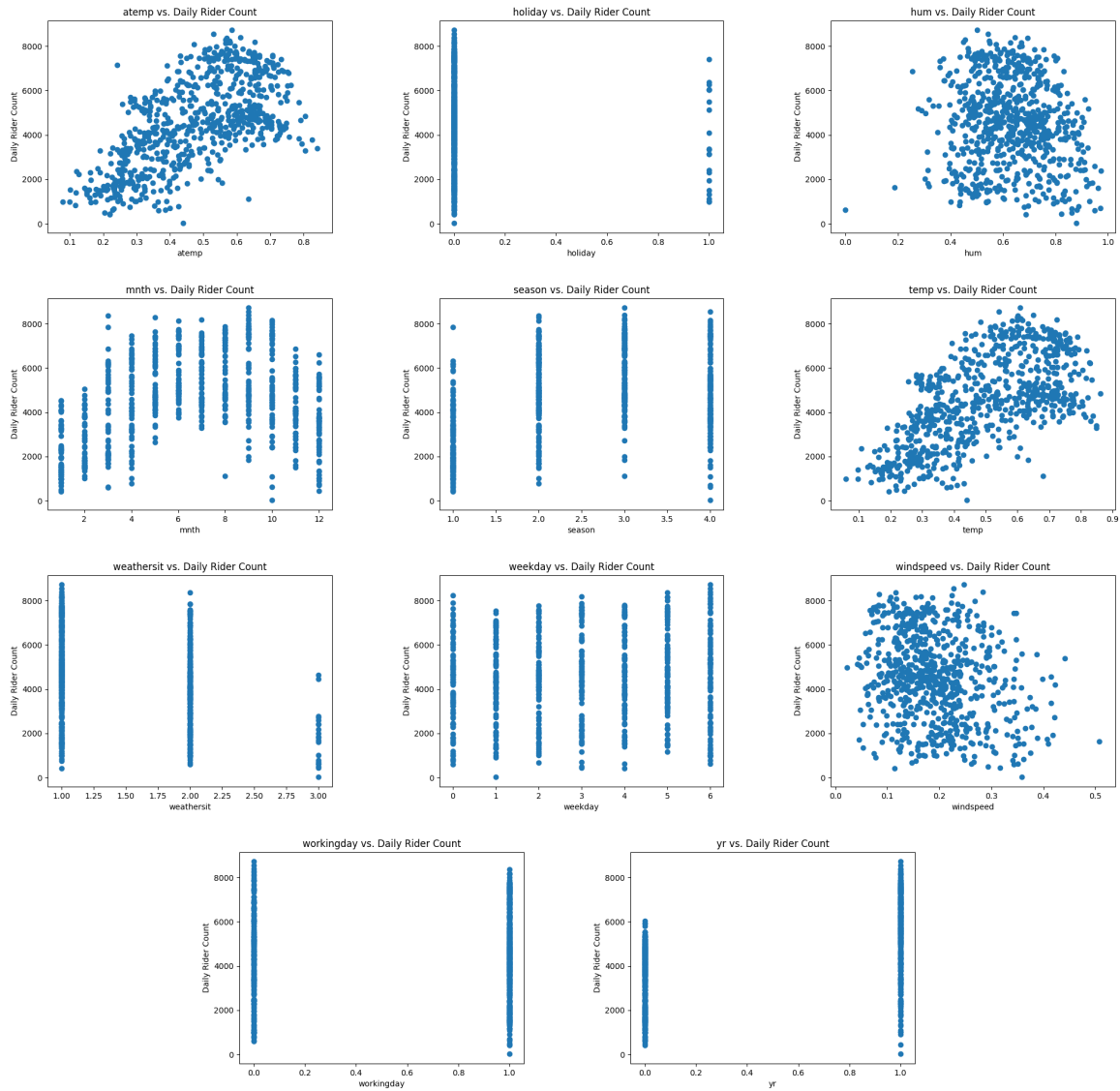
Solution

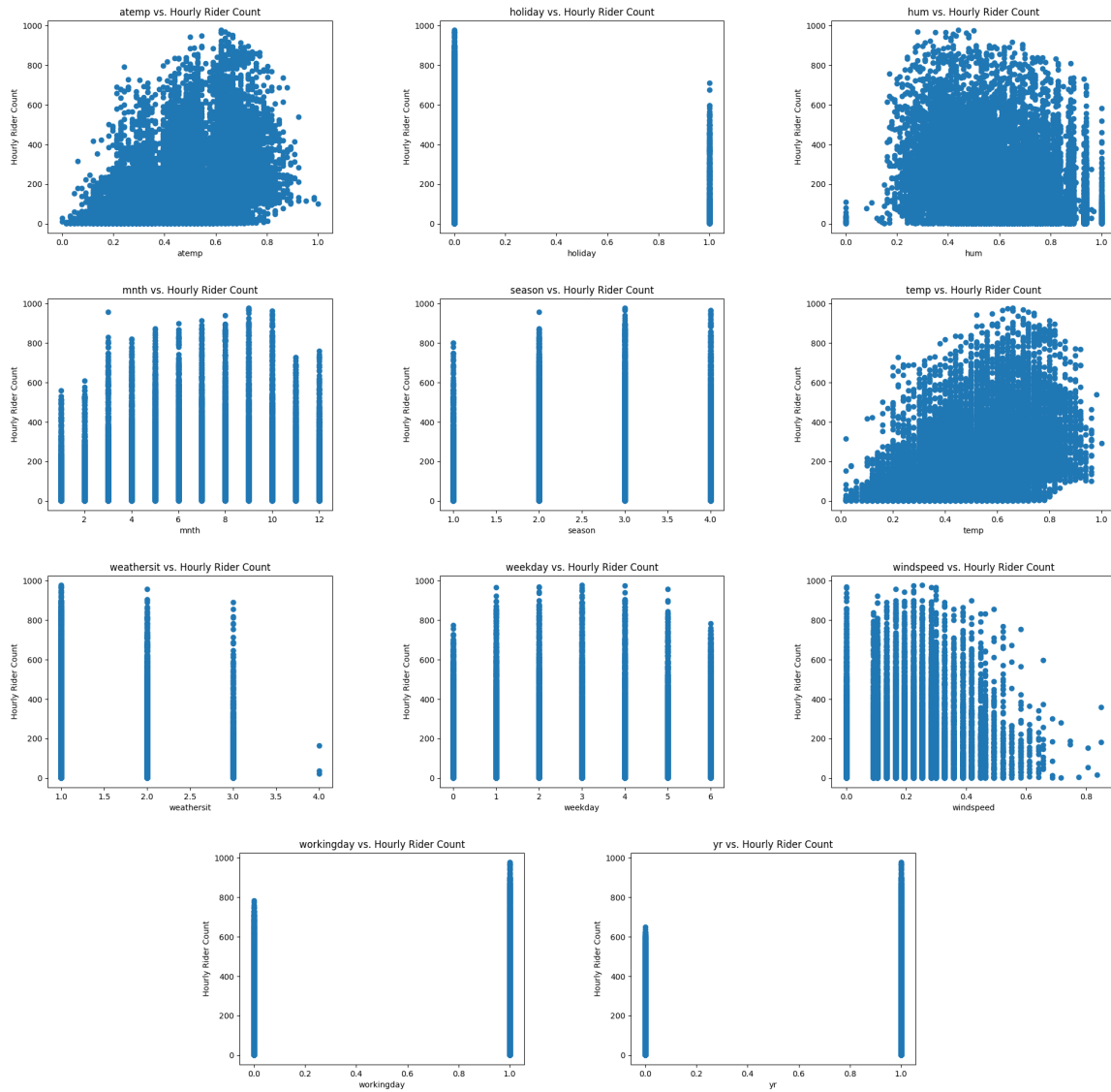
The necessary packages and files are imported into the Python script:

```
10 import pandas as pd
11 from matplotlib import pyplot as plt
12 from sklearn.linear_model import LinearRegression, Ridge, Lasso, BayesianRidge
13 from sklearn.model_selection import KFold
14 from sklearn.metrics import mean_squared_error
15 import numpy as np
16
17 hour = pd.read_csv("hour.csv")
18 day = pd.read_csv("day.csv")
19
20 cnt_hour = hour["cnt"].values.reshape(-1, 1)
21 cnt_day = day["cnt"].values.reshape(-1, 1)
```

The data is explored to examine its features. Both hourly and daily rider data is examined. The graphs outputted for the daily and hourly ridership are shown on the next pages, respectively.

```
1 for col in range(len(hour.columns)):
2     if hour.columns[col] not in ["instant", "cnt", "dteday", "registered", "casual"
3         ]:
4         plt.figure()
5         plt.scatter(hour[hour.columns[col]].values.reshape(-1, 1), cnt_hour)
6         plt.title("{} vs. Total Rider Hourly Count".format(hour.columns[col]))
7         plt.xlabel("{}".format(hour.columns[col]))
8         plt.ylabel("{}".format("Total Rider Count"))
9         plt.savefig("images/hour_{}.png".format(hour.columns[col]))
10        plt.clf()
11
12 for col in range(len(day.columns)):
13     if day.columns[col] not in ["instant", "cnt", "dteday", "registered", "casual"
14         ]:
15         plt.figure()
16         plt.scatter(day[day.columns[col]].values.reshape(-1, 1), cnt_day)
17         plt.title("{} vs. Total Rider Daily Count".format(day.columns[col]))
18         plt.xlabel("{}".format(day.columns[col]))
19         plt.ylabel("{}".format("Total Rider Count"))
20         plt.legend()
21         plt.savefig("images/day_{}.png".format(day.columns[col]))
22        plt.clf()
```





Based on the feature graphs, temperature was reduced to the best feature to model. According to the README the temperature was normalized with a 41 °C maximum so the temperature array is multiplied by 41 for human readability.

```
1 day_temp = day["temp"].values.reshape(-1, 1) * 41
```

KFold cross-validation was used with a 10-split. The regression models applied were linear, ridge, lasso, and bayesian through their respective sklearn packages. For each k-fold, the models are fitted using the separated training data.

```
1 crossvalidation = KFold(n_splits=10)
2 linear = LinearRegression()
3 ridge = Ridge()
4 lasso = Lasso()
5 bayesian = BayesianRidge()
6
7 for train_index, test_index in crossvalidation.split(day_temp):
8     X_train, X_test = day_temp[train_index], day_temp[test_index]
9     y_train, y_test = cnt_day[train_index], cnt_day[test_index]
10    linear.fit(X_train, y_train)
11    ridge.fit(X_train, y_train)
12    lasso.fit(X_train, y_train)
```

The test portion of the data was then used to test the prediction quality of the models. The mean square error was calculated between the test portion data and the values predicted by the models.

```
1 linear_pred = linear.predict(X_test)
2 ridge_pred = ridge.predict(X_test)
3 lasso_pred = lasso.predict(X_test)
4 bayesian_pred = bayesian.predict(X_test)
5
6 mse_linear = mean_squared_error(y_test, linear_pred)
7 mse_ridge = mean_squared_error(y_test, ridge_pred)
8 mse_lasso = mean_squared_error(y_test, lasso_pred)
9 mse_bayesian = mean_squared_error(y_test, bayesian_pred)
```

The following a table showing the different mean squared error values of the models.

Model	Mean Squared Error
Linear	4306257.419012716
Ridge	4306194.805960745
Lasso	4306018.563140404
Bayesian	4301889.80709847

